Dr. Bandana Priya

Assistant Professor

A N College, Patliputra University, Patna

Dr. Ganesh Kumar Thakur

Assistant Professor

Government Engineering College, Vaishali

CHYREN PUBLICATION

Palwal, Haryana-121102

Mob.:+91 9812453066, Tel.: +911275-455-202

E-mail: chyrenpublication@yahoo.com

https://www.chyrenpublication.com

ISBN 978-93-7143-814-8

9 789371 438148

Book Ref No.: 1676

The Future of Engineering Innovation: Smart Technologies and Advanced Design@2026 Volume-IIIrd

THE FUTURE OF ENGINEERING INNOVATION

Smart Technologies and Advanced Design

Dr. Bandana Priya

Dr. Ganesh Kumar Thakur

Volume IIIrd – 2026

The Future of Engineering Innovation: Smart Technologies and Advanced Design@2026 Volume-IIIrd

Dr. Bandana Priya
Dr. Ganesh Kumar Thakur

CHYREN PUBLICATION

Office: Rashulpur Road, Near Bijli Gate, Palwal, Haryana-121102
Reg. Office: 742, Ground Floor, Maruti Chowk, Gitorin, Delhi-30
Mob. +919812453031, 47, Tel: +911275-455-202
E-mail: chyrenpublication@yahoo.com
https://www.chyrenpublication.com

Publication: 1st Edition, 31/03/2026

ISBN: 978-93-7143-814-8
Price: Rs 599/-

Book Ref. No.-1676
Printed & Published By

CHYREN PUBLICATION

Palwal Haryana, India

Copyright ©2026 CHYREN PUBLICATION, *all rights* reserved. No part of this publication may be reproduced or transmitted in any form or by any means without permission of a publisher or individual author. Any person who does any unauthorized act in relation to this publication may be liable to criminal prosecution and civil claims for damages.

Disclaimer

The views expressed by the contributors in the Book Chapter published by Chyren Publication, Palwal, (HR) are their own. They do not necessarily reflect the views of the editor/publisher. The editor/publisher is not responsible in any way for any liability arising from the content/text of this book.

Table of Contents

CHAPTER-01 01
Deep Learning Models Incorporate Ethical Constraints by Design
Dr. Deena Nath Gupta
Mr. Pradeep Kumar Tiwari
Mohd Owais Ghazi
Dr. Pravesh Chandra Srivastava

CHAPTER-02……19
Event Handling in Java Programming
Mr. Rahul Senthiya
Prof. Archana Gupta
Mr. Rohit Chaudhary
Dr. Ranjan Walia

CHAPTER-03……44
Reinforcement Learning Based CPU and Memory Scheduling
Mr. Yogesh Kumar
Dr. Raghavendra Patil G E
Mr. Shikhar Prakash

CHAPTER-04……64
Neurophysiology of Movement Disorders
Dr. Hina Kauser

CHAPTER-05……87
Multithreading in Java
Ms. Rashim Rana

CHAPTER-06……103
Towards Immersive Classrooms: Practical Approaches to AR and VR Adoption in Indian Educational Institutions
Er. Rishabh Aryan

CHAPTER-07……121
Automated Multi-Class Chest X-Ray Classification: A Hybrid cGAN-Transfer Learning Framework for Detecting Normal, Tuberculosis, Pneumonia, and COVID-19 Severity Levels
Er. Rishabh Aryan

CHAPTER-08……136
Predicting Student Dropout and Satisfaction in Virtual Learning Environments: A Comparative Study of ML Algorithms.
Er. Rishabh Aryan

CHAPTER-09……146
Fundamentals of Computer Vision in Biomedical Contexts
Chandrapal Singh Arya
Dr. Sakshi Kumar
Mr. Tushar Gupta
Harjinder Singh

CHAPTER-10……159
Implementing AI in Healthcare
Ms. Arti Nagar
Dr. Mukta Makhija
Mr. Neerag kumar (Agarwal)
Mr. Ramachandra Turkani

CHAPTER-11……172
Normalisation in Database Management Systems
Prof (Dr.) Meenakshi Chaturvedi
Dr. Vishal Gupta
Dr. Rupinder Kaur
Dr. Sandeep Singh

Preface

Engineering has always been a driving force behind technological advancement and societal development. In the 21st century, the pace of innovation has accelerated significantly due to the integration of smart technologies, digital systems, and advanced design methodologies. The book *"The Future of Engineering Innovation: Smart Technologies and Advanced Design @2026 – Volume III"* aims to explore emerging trends, research developments, and practical applications that are shaping the future of modern engineering disciplines. This volume brings together scholarly contributions that highlight the growing role of intelligent systems, artificial intelligence, automation, advanced materials, and sustainable design in engineering practices. With industries rapidly adopting Industry 4.0 concepts, engineers today are expected to integrate data-driven technologies, smart sensors, and digital simulation tools into their design and development processes. Such transformations are redefining how mechanical, electrical, civil, and computer engineering systems are conceptualized, developed, and implemented. The chapters included in this volume focus on interdisciplinary approaches to solving complex engineering challenges. Researchers and practitioners have contributed to studies on smart manufacturing, intelligent mechanical systems, Internet of Things (IoT) applications, advanced structural design, and innovative computational techniques.

Editors

Acknowledgement

The completion of the book *"The Future of Engineering Innovation: Smart Technologies and Advanced Design @2026 – Volume III"* has been made possible through the support, encouragement, and valuable contributions of many individuals and organizations. We would like to take this opportunity to express our sincere gratitude to all those who played an important role in the development of this volume.

First and foremost, we extend our heartfelt appreciation to all the contributing authors whose research work, insights, and academic dedication have enriched the quality and depth of this publication. Their scholarly efforts and commitment to advancing engineering knowledge have made this volume a meaningful contribution to the field of smart technologies and advanced design. We would also like to thank the reviewers and subject experts who devoted their time and expertise to carefully evaluate the chapters. Their constructive suggestions, critical feedback, and academic guidance helped improve the clarity, accuracy, and overall quality of the content presented in this book. Our sincere thanks go to the academic institutions, research organizations, and professional bodies that continuously support research and innovation in engineering and technology. Their encouragement and academic environment inspire researchers and scholars to explore new ideas and contribute to technological development.

Editors

ISBN: 978-93-7143-814-8

Chapter - 1
Deep Learning Models Incorporate Ethical Constraints by Design

Dr. Deena Nath Gupta
Associate Professor (School of Engineering & Technology)
Maharishi University of Information Technology, Noida
Email: deenanath.gupta@muit.in

Mr. Pradeep Kumar Tiwari
Assistant Professor (Department of Computer Applications)
Dayanand Academy of Management Studies, Kanpur
Email-id- pradeeptripathi.iit@gmail.com

Mohd Owais Ghazi
Assistant professor (Department of Computer Applications)
Noida institute of engineering and technology
Email ID: owaisghazi2525@gmail.com

Dr. Pravesh Chandra Srivastava
Associate professor (Department of Applied science and humanities)
IIMT College Of Engineering, Greater Noida
Email-id- praveshchandra78@gmail.com

Abstract
Deep learning has revolutionized the science of artificial intelligence as it allows machines to obtain hierarchical data representations of raw inputs. The chapter gives an in-depth discussion of the two main classes of deep learning architecture, namely ANN Artificial Neural Networks., Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs) along with Long Short-Term Memory (LSTM) cells, Transformer networks, and Generative Adversarial Networks (GANs). Specifically, each architecture is addressed in terms of structural design, mathematical as well as practical capabilities. The original architecture diagrams are provided and demonstrated in details with aspects of data flow, component functions and distinguishing design principles. The chapter provides a summary of the historical empirically positive results

which include the discovery of AlexNet being the first model to provide results on ImageNet [3], the self-attention system in the Transformers [5], the generative adversarial training paradigm in GANs [6]. The training approaches are the backpropagation [12], dropout regularization [14] and batch normalization [13]. The trade-offs of architectures define which model to select and comparative analysis assists practitioners of other models. The applications of the computer vision, natural language processing, speech recognition, medical imaging, and generative AI are reviewed. The identification of the existing issues and future research prospects of model efficiency, interpretability and multimodal learning are presented at the end of the chapter.

Highly qualified undergraduate and graduate students, practitioners requiring to secure a methodical viewpoint on the current systems of deep learning could use this chapter.

Keywords: Deep Learning, Artificial Neural Networks, Convolutional Neural Networks, Recurrent Neural Networks, LSTM, Transformer, Self-Attention, Generative Adversarial Networks, Backpropagation, Representation Learning, Computer Vision, Natural Language Processing, Regularization, Transfer Learning, Model Architectures.

Introduction

Deep learning is a subfield of machine learning, which involves artificial neural networks that are executed in several processing layers to produce successively high-level properties in raw data [1]. The fact that it automatically discovers complex structure in large bodies of data has triggered paradigm shifts in virtually every kind of science and technology, such as medical diagnosis and autonomous car control in addition to the study of natural language and the production of creative works of art.

Traditional machine learning pipelines were manual in the way the features were created, tedious and vulnerable. The most significant advancement in the field of deep learning is that given sufficient data and compute, a hierarchically structured network can jointly discover useful feature representations as well as decision boundaries through end-to-end optimization [2]. It is end-to-end learning, which is free of the feature engineering bottleneck and much more adaptable to unknown conditions.

In 2012, the ImageNet competition came to jump-start the modern deep learning with the AlexNet by Krizhevsky et al. [3] that seemed to win the competition by a large margin through the use of a deep convolutional network trained on GPUs. It had demonstrated that scale both of data and model capacity had qualitatively new performances possible. As, we have observed a growth in the rate of architectural progress: exceedingly profound VGGNets [8], ResNet residual connections [7], two-directional language comprehension with BERT [15], and the generating miracles of GANs [6].

The chapter is organized in the following manner. Part 2 provides the mathematical principles common to all the deep learning models. In parts 3-7, every one of big architecture classes ANNs, CNNs, RNNs/LSTMs, Transformers, and GANs are defined and has architectural diagrams and descriptions. A comparative analysis is found in section 8. In Section 9, real world applications have been surveyed, and future directions and current challenges are described in Section 10.

Deep Learning: A Mathematical Introduction.

The Neuron and the Activation Functions.

The unit of any neural network is an artificial neuron, which computes a weighted sum of its inputs, a bias value, and the output of this value by using a nonlinear activation function:

y = f(wx + b)

In this form, where x [?] n is the input vector, w [?] n is the weight vector, b is the scalar bias, and f ( ) is an activation function, the activation functions are: the Rectified Linear Unit (ReLU: f (z) = max(0, z)), the sigmoid (f (z) = 1)/1+e-), or the hyperbolic tangent (tanh). The reason why ReLU function has become the default choice in the present networks is due to its efficiency in computations and the vanishing gradient issue [2].

Gradient Descent Backpropagation Backpropagation

Learning takes place when a loss function L(th) on the training set is optimized with th being the set of learnable parameters. Canonical optimization: The stochastic gradient descent (SGD) algorithm is referred to as such:

th - th [?] e * [?]thL(th)

where e is the learning rate. The gradients [?]thL may be computed using the backpropagation algorithm [12] an effective approach to computing

the error signals of the backpropagation using the chain rule of calculus. Both Adam and RMSProp adaptive optimizers maintain estimates of per-parameter learning rates, which in effect vastly improve the convergence rate.

Regularization Techniques

The deep networks are highly prone to overfitting due to high parameters. The most significant regularization methods are: (i) Dropout [14]: randomizing activations in training to discourage co-adaptation of neurons; (ii) Batch Normalization [13]: the normalization of the inputs to a layer across the mini-batch to stabilize training; (iii) L2 weight decay: massivevalues of weights discourage it; and (iv) data augmentation.
ANNs Artificial Neural Networks (ANNs).
Architecture
Artificial neural network consists of an input layer, one or more hidden layers, and an output layer with the layers being fully interconnected (Figure 1). Each connection has a weight to be learned and each neuron has a weight to learn. The output of layer l is:
al = f(Wlal-1 + bl)
where Wl and bl are the weight matrix and the bias of the l th layer. This architecture is also known as Multi-Layer Perceptron (MLP) or fully connected network.

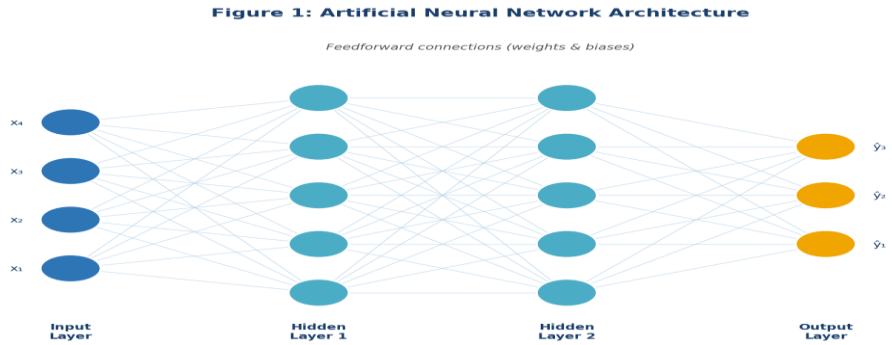


Figure 1: Architecture of an Artificial Neural Network showing input, hidden, and output layers with full connectivity. Each node represents a neuron applying a nonlinear activation function to the weighted sum of its inputs.

Diagram Explanation — Figure 1

Figure 1 is a feed forward ANN and the model has four inputs (x1-x4), two hidden data layers each with 5 neurons and three output data (y1-y3). The connection lines between the layers are also weighted connections: the edges have a scalar weight w that boosts the signal that neuron i is sending to neuron j. No recurrent or lateral connections i.e. the information flows in the left to right directions (feedforward networks). The two layers of the hidden form each represent a pair of cycles of representations transformation; the former hidden layer recognizes lower order abstractions in the input and the latter hidden layer combines them into higher order abstractions. In the case of multi-class classification, the softmax activation is used in the output layer and provides the distribution of the probability among the classes. All the weights are also optimized concurrently using backpropagation training to minimize the decrease in the cross-entropy error between projected probabilities and ground-truth labels [1,2,12].

Proprietary and Usages.

ANNs are approximators of any form of universal functionality: ANN with a single hidden layer is a universal approximator of any continuous function with arbitrary accuracy (Universal Approximation Theorem). In practice, more general networks are more parameter-efficient and have increased generalization ability. ANNs have found wide use in classification of tabular data, regression and elements of more literally focused networks. Their weakness is that they do not have inductive biases that can be applied to structured inputs like images (addressed by CNNs) or sequences (addressed by RNNs and Transformers) [9].

Convolutional Neural Networks (CNNs).

Architecture

CNNs were designed to make use of the grid-like category of the data (images, audio spectrograms, video frames) substituting fully connected operations with three specific layers: convolutional layers, pooling layers and fully connected layers (Figure 2).

A convolutional layer applies K learnable fxfxc sized filters (kernels) on the input features map. The filters are cut through the spatial dimensions and a dot product at every point is executed to produce an activation map. In the case of one filter w:

$S(i,j) = (I \ w)(i,j) = S[?]\ S[?]\ I(i+m, j+n) \ w(m,n)$

A count of parameters can be compared drastically with that of a fully connected layer through the use of parameter sharing (filter used at each spatial position) and translation equivariance is feasible.

Figure 2: Convolutional Neural Network (CNN) Architecture

Figure 2: CNN architecture pipeline for image classification. Input images pass through alternating convolution + ReLU and pooling stages for hierarchical feature extraction, followed by flattening and fully connected layers for classification.

Diagram Explanation — Figure 2

Figure 2 traces the whole process of a CNN whereby the raw pixels are initially processed to the possibilities of the classes. The input block (blue, left-hand) stores a 32x 32x3 image of RGB. The first convolutional layer that includes the ReLU activating (bright blue) is a 28x28 feature map with 32 feature maps representing a localized low level structure e.g. horizontal edges, color blobs or oriented textures. The second layer of max-pooling (mid-blue) with a factor of 2 downsampling to produce 14x14x32 maps, not only provides a factor of 2 computational reduction, but also local invariance to short translations.

The second convolutional block (dark blue) is based on 64 filters that produce more richer and higher-level features (10x10x64) and a second pooling reduces it to 5x5x64. The flatten operation (amber) quantizes the 3-D volume of feature to a 1-D 1600-value row, which is inputted to a fully connected (FC) layer of 256 neurons and ultimately a 10-way softmax predicting the classes. The arrow labels indicate that the size of the dimensions in space along with the count of depth (number of channels) are increasing in a design principle of CNNs that is also a

trade-off between spatial resolution and the representational richness of all the stages [3,8].

Landmark Architectures

The power of deep CNNs was demonstrated in 2012 with AlexNet [3] at ImageNet, the first network to use ReLU activations and dropout, and the first network to use them on a large scale. The performance was powerfully connected to the depth (only with the 3x3 kernels) of VGGNet [8]. Networks with more than 152 layers are possible with the skip connections of ResNet [7], and the phenomenon of residual learning:

F(x) := H(x) [?] x - output = F(x) + x

This redefinition makes identity mappings learning easier in layers, fixes degradation in very deep networks, and state-of-the-art performance on a wide range of vision tasks [7].

Recurrent LSTM and Neural Networks.

Recurrent Neural Networks

Sequences -text, speech, time-series -require architectures that are capable of modeling time-dependencies. The solution to this is given by Recurrent Neural Networks (RNNs) by maintaining a hidden state h[?] that is updated at time t by the input x[?] and the prior hidden state h[?]-1:

h[?] = tanh(W[?]h[?]-1 + W[?]x[?] + b)

W[?] and W[?] are shared, and they are used in all steps so that they will take arbitrarily short sequences. However, vanilla RNNs are extremely sensitive to vanishing and exploding gradients in the event that they are trained with the backpropagation through time (BPTT) on long sequences, and can only induce long-range dependencies [9]..

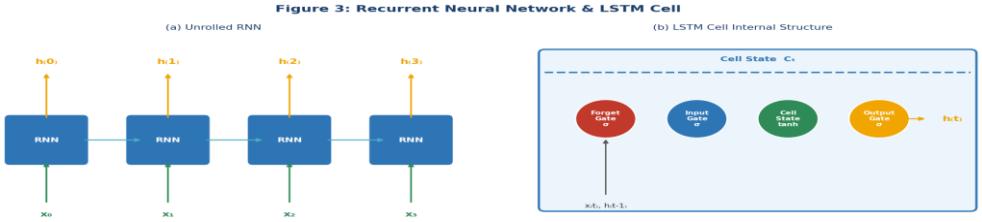


Figure 3: (a) Unrolled RNN across four time steps showing shared weights and hidden state propagation. (b) Internal structure of an LSTM cell with forget, input, cell state, and output gates controlling information flow.

Diagram Explanation — Figure 3

Panel (a) — Unrolled RNN: The four inputs x0 to x3 are passed through the four boxes of an identical RNN (they serve the same purpose, however, they share the same parameters). The system is fed with data represented by green upward arrows; the result of the hidden state is represented by amber upward arrows h0-h3. The horizontal blue arrows will push the hidden state h [?]-1 forward in time at this time. It is the case that the same weight matrices are re-used (weight tying) at each time, which brings a sort of time-consistency, and allows generalization to sequences of different length. Although that is seemingly so, all the four boxes are a single unrolled RNN cell which is visualized.

The LSTM Cell (b): Panel LSTM Cell The LSTM [4] solves the vanishing gradient problem by adding a cell state C[?] (the blue horizontal line along the middle of the dashed blue line at the top) as a linear information highway, or able to store the gradient of a long sequence of time periods. It possesses three gating mechanisms that support flow of information:

Forget Gate (s, red):The value of finding the fraction of the past cell state C[?]-1 to recall into values in (0,1) per element.

Input Gate (s, blue) + Candidate (tanh, green): The latter two give the definition of what one would like to be written to the cell state.

Output Gate (s, amber): Chooses the factors in the cell state that are to be unveiled as the disguised output h[?].

The gating has enabled the LSTMs to be competent at selective recalling and forgetting of data over hundreds of time steps, and as such, can be highly effective in long-term sequence assignments like machine translation, speech recognition and document classification [4].

GRU and Bidirectional RNNs versions.

The Gated Recurrent Unit (GRU), which is an extension of the LSTM, combines the forget and input gates into an update gate and does not maintain the cell state, cutting the number of parameters, but does not lose much performance. Bidirectional RNNs run sequences in both directions and concurrently and all outputs are given with the context caused by the past inputs and the future inputs- it can be used in activities like the named entity recognition and speech recognition [9].

Transformer Architecture Transformer Architecture Transformer Architecture is an architectural machine with a neural network focus, and based on neural networks.

Motivation and Overview

Although LSTMs are successful, they cannot calculate through time steps parallel to each other, and therefore create a bottleneck on long sequences. Then, Vaswani et al. [5] proposed the Transformer in 2017 and, in this case, the recurrence is absent, and a self-attention mechanism is applied, which implies that the positions of a sequence are related to each other in O(1) sequential operations.

Figure 4: Transformer Architecture (Encoder-Decoder)

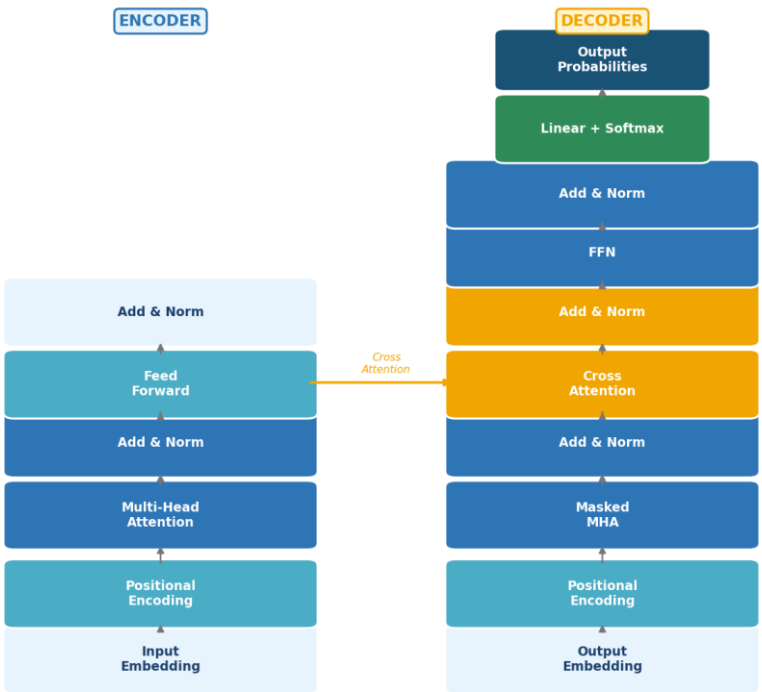


Figure 4: The Transformer encoder-decoder architecture. The encoder (left) converts input tokens into contextual representations; the decoder (right) generates output tokens autoregressively, using cross-attention to condition on the encoder output.

Diagram Explanation — Figure 4

Figure 4 illustrates the entire Transformer encoder-decoder of the initial sequence-to-sequence paradigm of the machine translation [5].

Encoder (left column): The tokens are converted to dense vectors with an embedding layer (light blue). The lack of a pre-determined position of Transformers renders the sinuoidal Positional Encoding (blue) which is required to provide the sequence order. The bulk of the encoder is composed of two or more layers of the same blocks with two sub-layers:

Multi-Head Attention: It uses multi-head scaled dot-product attention, where a query (Q), key (K) and value (V) are projected to multiple low-dimensional spaces: Attention(Q,K,V) = softmax(QK/[?])d[?])V. Multi-head This allows multi-head attention to multi-subspaces of representations.

Position-wise Feed-Forward Network (FFN): This is a two-layer MLP, which is applied at other positions on its own.

Add and Norm (residual connection followed by Layer normalization), which permit the free flow of the gradients across any depth, encases the two sub-layers [5].

Decoder (right column): This is also the case with the decoder where the output tokens are encoded similarly in which they are embedded and positional. The initial of its sub-layers is a Masked Multi-Head Attention, which does not permit positions to attend to future tokens in training (autoregressive property). The queries and keys of the decoder are fed to the critical Cross-Attention layer (amber boxes), at the encoder output side, in such a way that the decoder will only pay attention to the encoder states that are relevant in the generation of each output token. The amber cross-attention arrow to the encoder and the decoder is what describes this conditioning. This is ultimately followed by the Linear + Softmax, which makes a prediction of a vocabulary likelihood distribution.

BERT [15] simply uses the Transformer encoder and masked language modelling pretraining, which has state of art NLP performance. The GPT-series models [11] only require the decoder to produce autoregressive texts. It has been demonstrated with Vision Transformers (ViT) that the same architecture used on image patches could achieve a competitive vision performance.

Complexity and Scalability

Self-attention mechanism has a complexity that is O(n2d) in sequence length n, which can be expensive in cases of sequences that are very long. Such variants of Transformers as Sparse Transformers, Longformer,

Flash Attention, and Transformers can be applied in genomics, document understanding, and long-form audio to overcome the limitations [5,11].

Generative Adversarial Networks (GANS)

Architecture and Training Objective.

Goodfellow et al. [6] introduced generative models in 2014 as generative models, trained to produce samples of data, which are realistic through learning as a two-player minimax game between Generator G and Discriminator D.

minG maxD V(D,G) = [log D(x)] + [log(1 [?] D(G(z)))]

A random noise signal z pz(z) is inputted into the generator G, which produces a synthetic data sample G(z). D is an actual sample fed on a real sample x pdata or a counterfeit sample G(z). G is conditioned to induce D to commit a mistake (to deceive it into thinking that the fakes are real) and D is conditioned to make the appropriate judgment between the real and the fake.

Figure 5: Generative Adversarial Network (GAN) Architecture

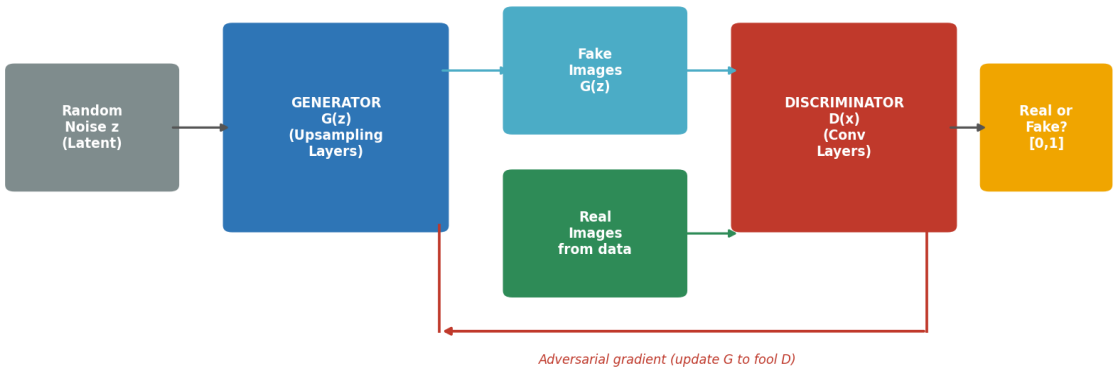

Random Noise z (Latent)
GENERATOR G(z) (Upsampling Layers)
Fake Images G(z)
Real Images from data
DISCRIMINATOR D(x) (Conv Layers)
Real or Fake? [0,1]
Adversarial gradient (update G to fool D)

Figure 5: GAN architecture showing the adversarial training loop. The Generator maps random noise to synthetic samples; the Discriminator classifies real vs. fake inputs. Gradient signals from the Discriminator update the Generator to progressively improve sample quality.

Diagram Explanation — Figure 5

All the training process of GAN is outlined in figure 5. The latent space is modeled as grey block on the left: the noise z is a random sample of normal distribution of Gaussian. The Generator (blue block), which is a

deep upsampling network (applying transposed convolutions or progressive upsampling) that encodes z into a synthetic image G(z) is fed the input of this vector (teal block).

The Generator provides two streams to the Discriminator (teal arrow) fake images and real images respectively. It provides one probability rank (amber block) indicating the presence or absence of any of the inputs that is genuine or faked. The adversarial gradient (it is the red arrow) is pointing in the opposite direction, i.e. the loss of the Discriminator is being differentiated with regards to the parameters of G to provide it with a learning signal on how G can better simulate the appearance of its outputs.

This creates the effect of such a competitive situation where G produces samples that are no longer distinguishable to the actual data distribution. Nash equilibrium implies that G is an ideal representation of p data and D assumes the value of 0.5 in every sample. In practice, the training stability requires some tricks: progressive enlargement, spectral normalization, Wasserstein distance (WGAN), and gradient penalty [6].

Variants and applications of GAN..

DCGAN DCGAN trains the two networks with convolutional layers in each of the networks (G and D) to produce photorealistic images. cGAN Conditional GAN DCGAN Conditional GAN trains the two networks using a class label to enable controlled synthesis. StyleGAN is a face generation architecture that is composed of a style-based generator, which produces images of unparalleled quality. CycleGAN image-to-image translation (e.g. horse - zebra) is made without using paired training images. They may be applied in super-resolution, data augmentation, generation of drug molecules and creative media [6].

Architectural Comparisons of Deep Learning.

Table 1 proposes a side-by-side comparison of the main six deep learning architectures that have been presented in this chapter, including presenting their primary application, major operations, their key strengths, and constraints.

Model	Primary Task	Key Operation	Strength	Limitation	Ref

Model	Primary Task	Key Operation	Strength	Limitation	Ref
ANN	Classification	FC Layers	Universal approximator	No spatial/temporal	[1,2]
CNN	Image tasks	Convolution	Local feature extraction	Large data needed	[3,7,8]
RNN/LSTM	Sequences / NLP	Recurrent loop	Long-range dependencies	Vanishing gradient	[4,9]
Transformer	NLP, Vision	Self-attention	Parallelism, long context	Quadratic complexity	[5,15]
GAN	Generation	Adversarial game	Realistic synthesis	Training instability	[6]
ResNet	Deep vision	Skip connections	Very deep networks	Complex tuning	[7]

Table 1: Comparative summary of major deep learning architectures. References indicate primary source papers.

The principle of architectural bias is one of the main themes that can be elicited out of this comparison: there is an inductive bias in each model, which is based on the target data modality. CNNs The translation invariance of spatial information RNNs The sequential order of temporal information Transformers The global context translated by attention GANs A generative prior. The structure selection process is divided into mapping the inductive bias of the model into the form of the problem [2,9].

Transformers and CNNs can also be trained at a large scale by being computationally highly parallelizable on the current hardware,

especially on GPU/TPU hardware. RNNs are sequential. The dynamism of the minimax criterion is infamously difficult to train GANs. The skip connection in the ResNet style has proven to be a powerful and useful universal component to have in nearly all modern designs [7].

Applications had been done using Deep Learning Models.

Computer Vision

Uses CNNs and vision transformers are used in applications such as object detection (YOLO, Faster R-CNN), semantic segmentation (U-Net, DeepLab), face recognition, medical image (tumor detection in MRI/CT scans), and autonomous vehicle perception. The production vision systems continue to rely on backbones based on ResNet [3,7].

NLP (Natural Language Processing).

The new masters of NLP are transformers: BERT [15] proves itself state of the art on question answering, sentiment analysis and named entity recognition when trained on masked pre language models. Text, text summarization, code completion, as well as few-shot learning are also well produced by GPT-family models [11]. RNNs/LSTMs can be used in high-resource limited systems and real-time inference.

Speech and Audio

The basis of the Automatic Speech Recognition (ASR) systems (e.g. DeepSpeech, Whisper) is the RNNs and Transformers. The CNNs are trained on mel-spectrogram features; sequence models are trained on the acoustic features to text. Neural vocoders and waveNet (dilated causal CNNs) are models that are used in generating speech in a natural way.

Healthcare and Drugs Discovery.

Medical imaging diagnostics Medical imaging diagnostics CNN-based systems have shown impressive results and perform on par with radiologists on dermatology, ophthalmology and radiology tasks. Accelerating drug discovery with the help of Molecular graph Neural Networks, and Transformers, or predicting the shape of proteins (AlphaFold) and designing new candidate molecules is possible [1,2].

Generative AI

GANs and diffusion models (a deep learning implementation plus stochastic differential equations) can be used to generate images of high fidelity, videos, synthesize 3-D shapes, and transfer styles. Large Language Models (LLMs) offer multi-turn conversation, code generation

as well as complex reasoning, and operate on Transformer architecture including GPT-4 [11].

Current issues and Future outlook.

Explainability and Interpretability.

Deep networks remain largely black boxes: one cannot easily tell why this or that prediction was made. The concept of gradient-weighted Class Activation Maps (Grad-CAM) and SHAP, the use of LIME and attention visualization methods, are partially the answer to the problem, but a conceptualized approach to mechanistic interpretability is an unresolved research topic. Healthcare and legal decision-making are high stake domains that need to be explained [2].

Generalization and Data Efficiency.

The available deep learning systems typically require massive labeled data. The self-supervised learning, few-shot learning, and data augmentation strategies are the self-supervised strategies aimed at mitigating this dependency. One critical unsolved problem is the generalization gap, in-distribution models that are good on distribution shift [10].

Sustainability and Performance.

The massive models (GPT-4, PaLM) are extremely time-consuming to train, which puts a question of power and carbon footprint. At present, model compression (pruning, quantization, knowledge distillation), neural architecture search (NAS), and specialized hardware (TPUs, neuromorphic chips) are all going on, aiming to make deep learning more efficient [2].

Robustness and Security

The deep learning models are vulnerable to adversarial example imperceptible perturbations of the input causing radical variation in the output, and to a change in the distribution on deployment. Certified robustness training, randomized smoothing training, and adversarial training partially defend. This is among the open issues of the safety-critical systems [9].

Multimodal and Embodied AI

Integration of vision, language, audio and embodied action in single models is the best frontier. The foundation models (CLIP, Flamingo, GPT-4V) exhibit emergent cross-modal reasoning and are trained on

multimodal data of large scale. Embodied AI applies deep learning to robotic controlling and robotic navigating and requires a tight integration of perception and control [11].

Conclusion

This chapter has provided a guided excursion of the significant deep learning designs that describe the state of the art in artificial intelligence. We have begun with the mathematics behind simple neurons, simple fully connected ANNs, and gone through the history of simple ANNs up to the spatial processing capabilities of CNNs, the temporal processing of RNNs/LSTMs, the global attention paradigm of transformers, and the adversarial generative paradigm of GANs. Each model was presented using architectural diagrams which had been scrutinized in detail to narrate information flow, role of components and design justification.

The comparative analysis revealed that none of the architectures are always superior to the rest, but both of them have relevant inductive biases, which is why they are especially appropriate to particular data modalities and task structures. In the selection of the architectures, the practitioners must take into account the tasks demands, the available information, the computational budget and interpretability demands. Deep learning systems continue to evolve at a fast pace, and hybrid architectures with convolutional and attention modules, diffusion models and neural ODEs are the future of the discipline.

As more systems are sensitive to society, such as health care, law, education, infrastructure, etc., in which deep learning can be used, interpretability, robustness, fairness, and efficiency are of utmost concern. The mathematical and architectural backgrounds of this chapter provide the conceptual means of equipping oneself with to understand not only the existing systems but also contribute to their responsible evolution..

References

[1] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436–444. https://doi.org/10.1038/nature14539

[2] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press. http://www.deeplearningbook.org

[3] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems, 25, 1097–1105.

[4] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[5] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. Advances in Neural Information Processing Systems, 30.

[6] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial nets. Advances in Neural Information Processing Systems, 27.

[7] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 770–778.

[8] Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015).

[9] Schmidhuber, J. (2015). Deep learning in neural networks: An overview. Neural Networks, 61, 85–117. https://doi.org/10.1016/j.neunet.2014.09.003

[10] Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. IEEE Transactions on Pattern Analysis and Machine Intelligence, 35(8), 1798–1828.

[11] Brown, T. B., Mann, B., Ryder, N., et al. (2020). Language models are few-shot learners. Advances in Neural Information Processing Systems, 33, 1877–1901.

[12] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. Nature, 323(6088), 533–536.

[13] Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. Proceedings of the 32nd International Conference on Machine Learning, 448–456.

[14] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 15(1), 1929–1958.

[15] Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. Proceedings of NAACL-HLT 2019, 4171–4186.

****

ISBN: 978-93-7143-814-8

Chapter - 2
Event Handling in Java Programming

Mr. Rahul Senthiya
Assistant Professor (Department of CSE)
Indraprastha Engineering College
Email id-rdsenthiya@gmail.com

Prof. Archana Gupta
Assistant Professor (Department of Computer Science)
Institute of Management Studies, Ghaziabad, University Courses Campus
Email-id- archana.gupta@imsuc.ac.in

Mr. Rohit Chaudhary
Assistant Professor (Department of Computer Science and Engineering)
Noida Institute of Engineering and Technology, Greater Noida, UP, 201306
Email ID - chaudharymtech@gmail.com

Dr. Ranjan Walia
Professor (**Department** of Computer Science and Engineering)
Chandigarh University, Mohali, Punjab (INDIA)
Email- ranjanwalia@gmail.com

ABSTRACT
Java Event handling is one of the basic paradigms of Java programing which allows an application to react dynamically to user inputs and to signals generated by the system. This chapter gives a detailed discussion of event processing in Java, starting with domains of conceptual aspects of event-driven programming, delegation event model, event classes and listener interfaces, anonymous inner classes, lambda expression and adapter classes, and the practical implementation of event processing by a GUI based on Swing and JavaFX toolkit. Other high-level aspects covered in the chapter are custom events, propagation of event and tips on how to construct strong and responsive applications. With the aid of

illustrative diagrams, comparative tables, and annotated code examples, the material is meant to provide readers with the theoretical knowledge as well as practical skills of Java event handling. The main points of contact are made between the classical event-handling techniques and the current lambda-based techniques with pitfalls being outlined together with the suggested solutions.

Keywords: Event Handling; Event-Driven Programming; Java Delegation Event Model; ActionListener; Swing; JavaFX; Lambda Expressions; GUI Programming; Event Listeners; Observer Pattern

INTRODUCTION

The modern software systems are not so passive because they are required to react to all stimuli given by the users, peripheral hardware, networks and timers. This reactive style of programming has come to be referred to as event-driven programming- a programming paradigm that has become central to the graphical user interface (UI), mobile applications, server solutions and even real-time applications. Java programming language has never been a sceptic of this paradigm with the rich and type-safe event-handling architecture that are grounded on clearly defined interfaces and class hierarchies.

The Java event model has an educative development. The events in the JDK 1.0 were through inheritance: responding to the user interaction required a particular object to over-ride certain methods. Even though this is easy, it led to the extreme integration of the elements of user-interfaces and application logic. The Delegation Event Model (DEM) was announced by Sun Microsystems immediately after the release of JDK 1.1, which is a decoupling of event sources and event handlers through listener interfaces [1]. The model is still the foundation of the Java event system to this day.

Later Javas added functionality to the event-handling environment. Java 5 introduced generics and annotations, and allowed many more type-safe event architectures. Java 8 lambda expressions eliminated much of the boilerplate of the single-method listener interfaces and the code is much easier and shorter to write and read [2]. The toolkit since Java 11, JavaFX, which replaced Swing as the preferred GUI toolkit, also

introduced property binding and observable streams of values as other event-handling tools [3].

The developments are arranged in this chapter in a sequential pedagogical sequence. Section 2 reviews fundamental concepts. Section 3 dissects the Delegation Event Model. Sections 4 through 7 examine specific listener types, adapter classes, inner classes, and lambda expressions. Section 8 covers GUI-level event handling in Swing and JavaFX. Section 9 introduces custom events, and Section 10 discusses best practices and common pitfalls. The chapter closes with a summary, discussion questions, and a reference list.

FUNDAMENTALS OF EVENT-DRIVEN PROGRAMMING

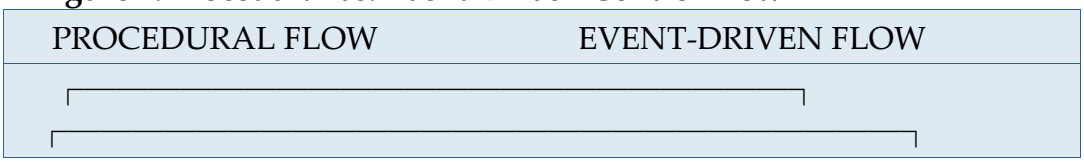
What Is an Event?

An event is a signal that something noteworthy has occurred. In a GUI context, events typically correspond to user actions such as clicking a button, typing a keystroke, moving the mouse, or selecting an item from a list. System events may also arise from timers firing, network data arriving, or hardware state changes. Abstractly, an event encapsulates three pieces of information: the source (the object that originated the event), the type (the category of occurrence), and any associated state (e.g., coordinates of a mouse click or the character produced by a key press) [4].

Event-Driven vs. Procedural Programming

In procedural programming, control flow is dictated by the program itself: one statement follows another in a predetermined sequence. In event programming, the stimuli determine the flow of control. The program can state its interests in particular events and provide handlers, also called callbacks that are invoked by the runtime when such event takes place. This inversion of control (also referred to as the Hollywood Principle: do not call us, we will call you) is the trait of event-driven design [5].

Figure 1: Procedural vs. Event-Driven Control Flow



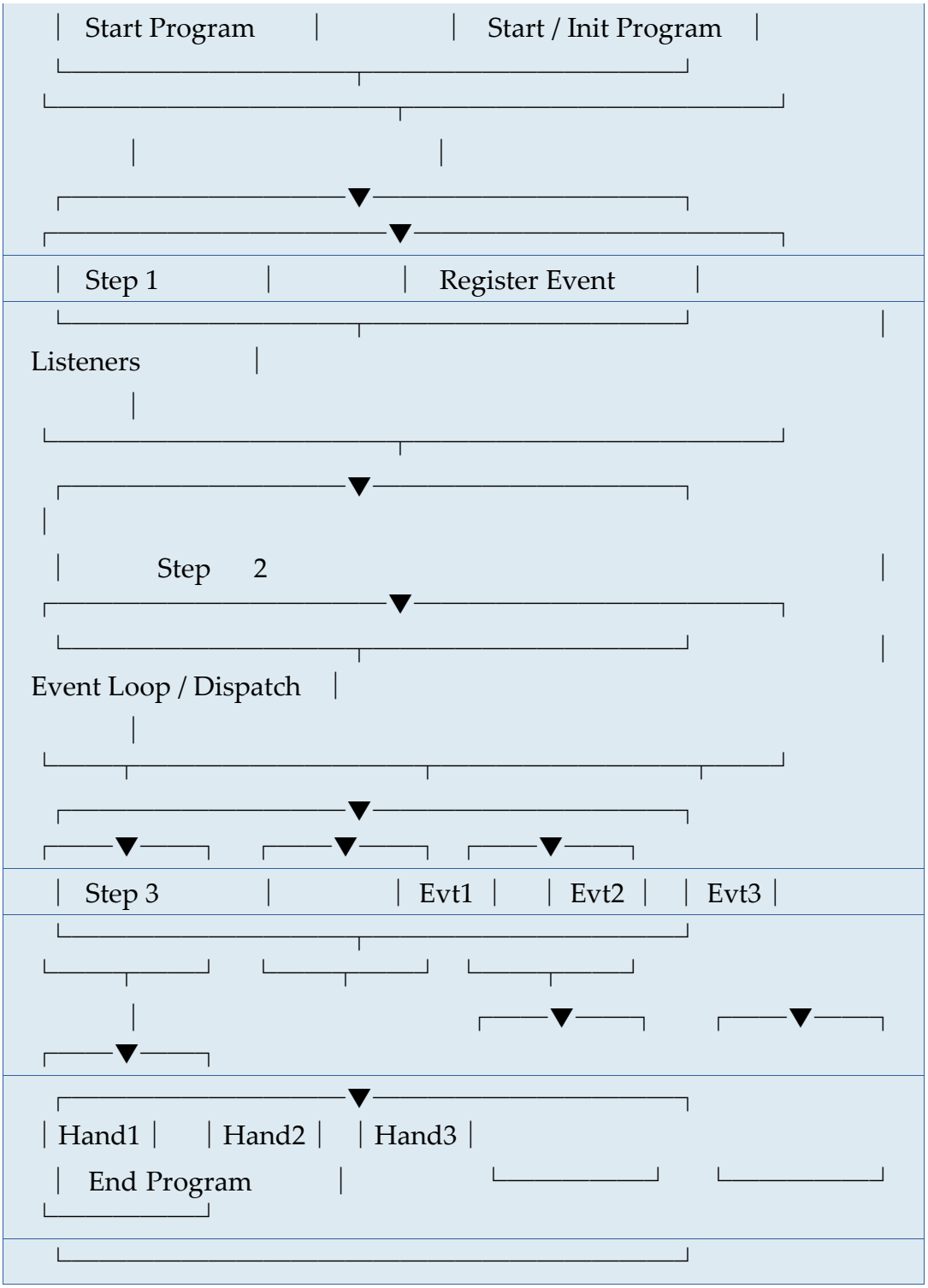

Start Program
Start / Init Program
Step 1
Register Event
Listeners
Step 2
Event Loop / Dispatch
Step 3
Evt1
Evt2
Evt3
Hand1
Hand2
Hand3
End Program

Figure 1. Procedural programs follow a fixed linear trail, and event driven programs respond to asynchronous impulses sent via an event loop to called handler functions.

The Observer Design Pattern

Java event model is a basic realization of the Observer (or Publish Subscribe) design pattern in the Gang of Four definition [6]. This is a type of pattern whereby Observers subscribe to one or more Subjects. When any Change is made in the State of the Subject, all the registered Observers are informed of it. The event source in the event system Java is the Subject and the other is the Observer. The result of this decoupling is loose code that is easy to maintain.

THE DELEGATION EVENT MODEL

Java 1.1 has introduced the Delegation Event Model (DEM): the model provides an organization of event handling around three collaborative roles, following the three cooperating roles as Event Sources, Event Objects, and Event Listeners [1].

Event Sources

Any object of Java capable of producing events is termed as an event source. The AWT/ Swing GUI toolkit has event sources such as JButton, JTextField and JCheckBox. Sources maintain a list of the subscribing listeners and notify the listeners when the events occurred. Registration is made in a normal form: addXxxListener to register and removeXxxListener to deregister.

Event Objects

Every event is represented by an instance of a class that extends java.util.EventObject. This base class records the source of the event via a getSource() method. Subclasses in the java.awt.event and javax.swing.event packages add further state: ActionEvent carries an action command string, MouseEvent carries pixel coordinates and button identifiers, KeyEvent carries key codes and character values, and so forth [7].

Event Listeners

A listener is an object that implements a specific listener interface, which declares one or more callback methods. When an event fires, the event source iterates through its list of registered listeners and invokes the

appropriate callback on each one, passing the event object as an argument. Because listener interfaces extend java.util.EventListener, the type system ensures that only objects implementing the correct interface can be registered for a given event category.

Figure 2: The Java Delegation Event Model

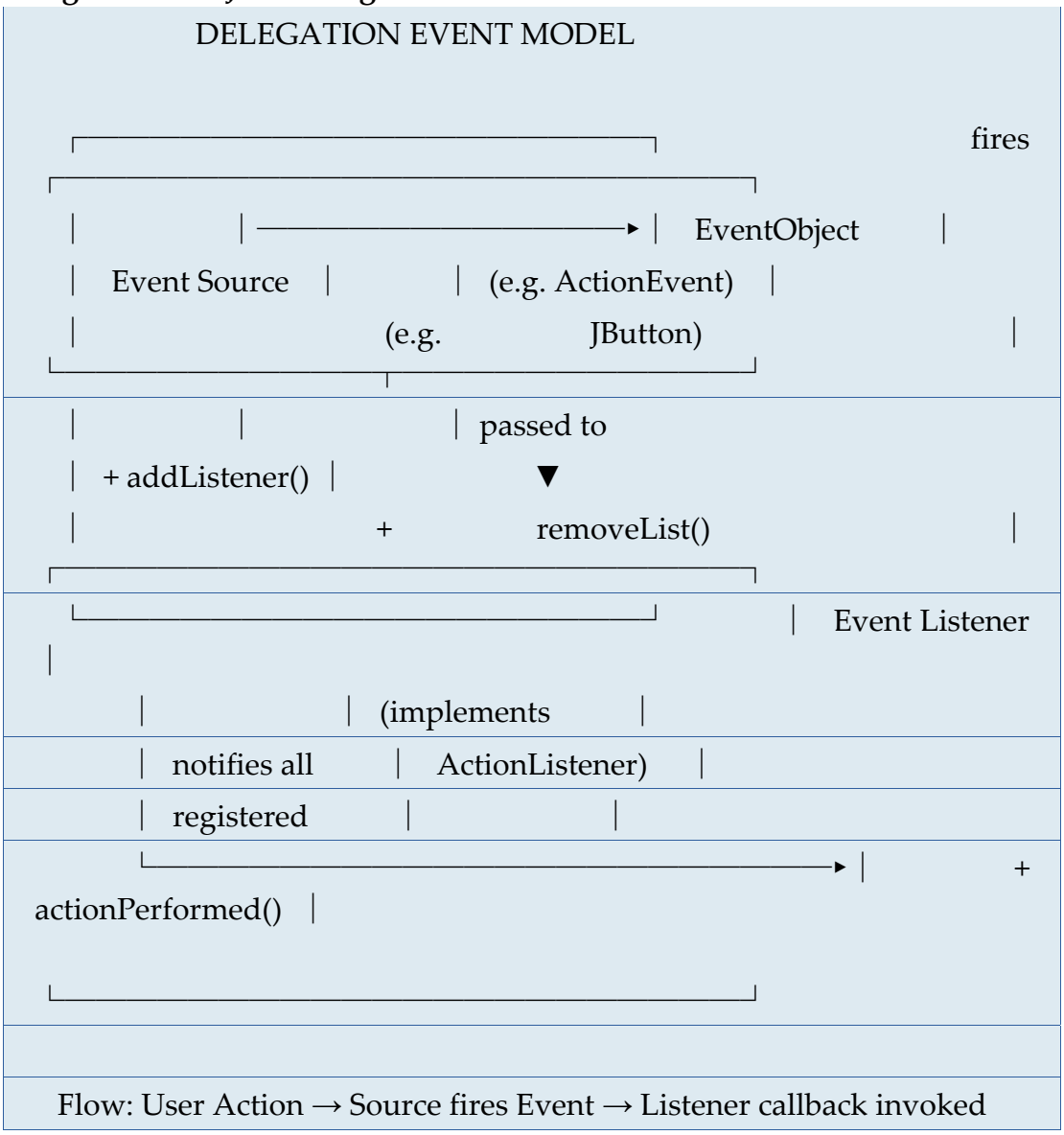


Figure 2. The three-role structure of the Delegation Event Model: the source fires an event object that is dispatched to registered listener callbacks.

Table 1: Common Java Event Classes and Their Source Components

Event Class	Package	Typical Source	Key Methods
ActionEvent	java.awt.event	JButton, JMenuItem	getActionCommand(), getModifiers()
MouseEvent	java.awt.event	Any Component	getX(), getY(), getButton()
KeyEvent	java.awt.event	Any Component	getKeyCode(), getKeyChar()
WindowEvent	java.awt.event	JFrame, JDialog	getWindow(), getNewState()
ItemEvent	java.awt.event	JCheckBox, JComboBox	getItem(), getStateChange()
FocusEvent	java.awt.event	Any Component	isTemporary(), getOppositeComponent()
ChangeEvent	javax.swing.event	JSlider, JSpinner	getSource()
ListSelectionEvent	javax.swing.event	JList, JTable	getFirstIndex(), getLastIndex()

Table 1. A selection of Java AWT and Swing event classes, their packages, typical source components, and the key methods used to interrogate event state.

EVENT LISTENER INTERFACES

Java offers a wider range of listeners interfaces, and all of them are related to types of events. The interfaces of any listener are all extensions

of the type of interface that is itself a marker interface (that is, it has no methods) and which is a common type of all the listeners. Concrete listener interfaces specify one or more methods which are called back named after the event that they all contend with.

ActionListener

The interface that is probably most widely used is the action listener. It provides one procedure, actionPerformed(ActionEvent e) that is called each time the action that it is bound to is invoked, usually a button press, menu choice, or press of the Enter key in a text field [7]. The following example demonstrates the canonical pattern for registering an ActionListener on a JButton:

```
import javax.swing.*;
import java.awt.event.*;

public class ButtonDemo extends JFrame {
    public ButtonDemo() {
        JButton btn = new JButton("Click Me");

        // Register an ActionListener using an anonymous class
        btn.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                System.out.println("Button clicked!");
            }
        });

        add(btn);
        pack();
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setVisible(true);
    }
```

```
    public static void main(String[] args) {
        new ButtonDemo();
    }
}
```

Code Example 1: Registering an ActionListener via an anonymous inner class on a JButton component.

MouseListener and MouseMotionListener

MouseListener declares five callbacks: mouseClicked(), mouseEntered(), mouseExited(), mousePressed(), and mouseReleased(). MouseMotionListener adds mouseDragged() and mouseMoved() for tracking movement. A single component can have multiple listeners of the same or different types registered simultaneously, and all are notified in the order of registration [8].

KeyListener

KeyListener provides three callbacks: keyPressed(), keyReleased(), and keyTyped(). The keyPressed() and keyReleased() methods report physical key events using key codes (virtual-key constants defined in the KeyEvent class), while keyTyped() reports the resulting Unicode character. For components that do not receive keyboard focus by default, setFocusable(true) must be called [7].

WindowListener

WindowListener monitors the lifecycle of top-level windows and declares seven callbacks: windowOpened(), windowClosing(), windowClosed(), windowIconified(), windowDeiconified(), windowActivated(), and windowDeactivated(). The windowClosing() callback is commonly used to intercept the user's attempt to close a window, allowing the application to prompt for confirmation or perform cleanup [9].

Table 2: Key Listener Interfaces, Their Methods, and Typical Use Cases

Listener Interface	Callback Methods	Typical Use Case
ActionListener	actionPerformed(ActionEvent)	Button clicks,

		menu selections, Enter in text field
MouseListener	mouseClicked/Pressed/Released/Entered/Exited	Click detection, hover highlighting, drag initiation
MouseMotionListener	mouseDragged(MouseEvent), mouseMoved(MouseEvent)	Drag-and-drop, tooltip tracking, cursor change
KeyListener	keyPressed/Released/Typed(KeyEvent)	Hotkeys, input validation, custom keyboard shortcuts
WindowListener	windowOpened/Closing/Closed/Iconified/...	Cleanup on close, minimize/maximize handling
FocusListener	focusGained(FocusEvent), focusLost(FocusEvent)	Field validation on blur, highlighting focused widget
ItemListener	itemStateChanged(ItemEvent)	Checkbox/radio toggle, combo-box selection

ChangeListener	stateChanged(ChangeEvent)	Slider value change, spinner change, tab selection

Table 2. A comparative overview of frequently used Java listener interfaces, listing their callback methods and the scenarios where each is most commonly applied.

ADAPTER CLASSES

Many listener interfaces, such as MouseListener and WindowListener, declare multiple methods. A class implementing such an interface must provide an implementation for every declared method, even those of no interest, resulting in numerous empty method bodies that clutter the code. To overcome this inconvenience, the java.awt.event package offers adapter classes: abstract classes which implement each of the listener interfaces and offer empty default implementations of every method [10].

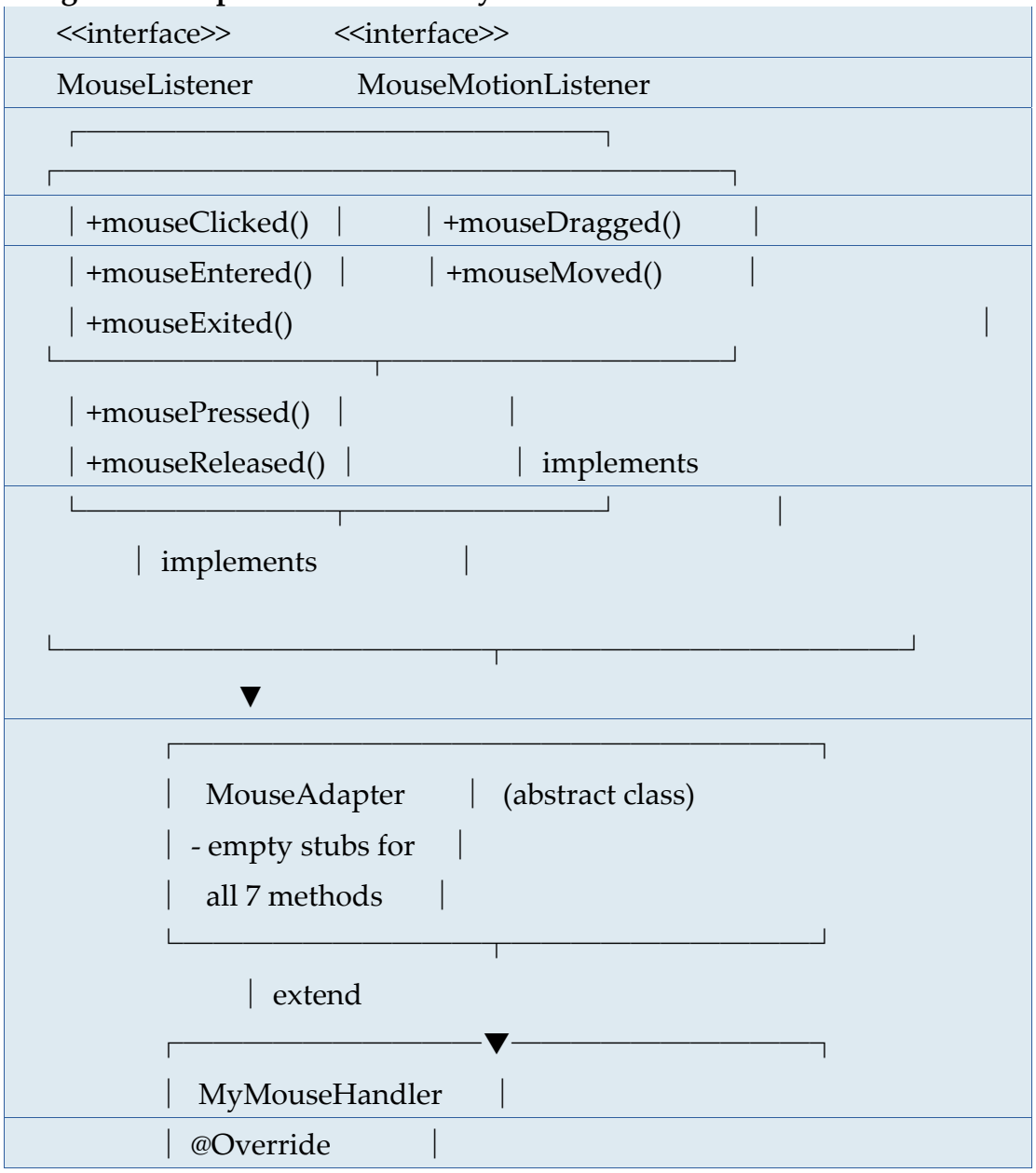
With an extension, an adapter class is used rather than direct implementation of the interface, and all a programmer has to do is to override the methods of interest. An example of this, MouseAdapter, is a mouse listener and a mouse motion listener giving the option to the child class that it will only handle mouseClicked without the need to explicitly handle other four mouse events.

```
// Using MouseAdapter to handle only mouseClicked
component.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {
        if (e.getClickCount() == 2) {
            System.out.println("Double-clicked at: " + e.getPoint());
        }
    }
```

```
    // mouseEntered, mouseExited, mousePressed, mouseReleased
    // are inherited as empty stubs — no need to implement them
});
```

Code Example 2To make only the mouseClicked() callback function, and have the rest of the four methods of MouseListener just inherited stubs.

Figure 3: Adapter Class Hierarchy for Mouse Events



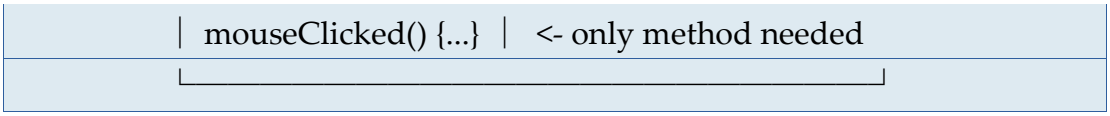


Figure 3. MouseAdapter is a stub-based adapter of both MouseListener and MouseMotionListener, and thus a subclass just needs to override those stubs that it needs.

INNER CLASSES AND ANONYMOUS CLASSES

Named Inner Classes

A class which is inner (or inner) (a class within a class) has the privilege of accessing the fields and methods of an outer class, and it is natural enough to place event-handling logic there which is necessary to communicate with the state of the outer GUI component. The advantage of named inner classes reusability is the possibility to give the same listener to other components [11].

Anonymous Inner Classes

Anonymous inner class This is a class that is defined and implemented upon usage without being named. It offers the least possible capability of in-lining the implementation of a listener. Java 1.1 has the idiomatic form of one-off listeners, anonymous inner classes, and the practice of writing one-off listeners has been discovered all through Swing code bases ever since.

Lambda Expressions (Java 8+)

All functional interfaces—interfaces declaring exactly one abstract method—can be implemented by a lambda expression in Java 8 and later [2]. Because most event listener interfaces (ActionListener, Runnable, ChangeListener) are functional interfaces, lambda expressions replace anonymous inner classes with considerably less syntactic overhead. The compiler infers the target interface from context, making the code more concise and readable.

```
// Anonymous inner class (pre-Java 8 style)
button.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
```

```
        label.setText("Clicked via anonymous class");
    }
});

// Lambda expression (Java 8+ style)
button.addActionListener(e -> label.setText("Clicked via lambda"));

// Method reference (even more concise when method signature matches)
button.addActionListener(this::handleButtonClick);
```

Code Example 3: Equivalent ActionListener registrations using an anonymous inner class, a lambda expression, and a method reference—illustrating the progressive reduction in verbosity.

Table 3: Comparison of Event Listener Implementation Styles

Criterion	Named Inner Class	Anonymous Inner Class	Lambda Expression	Method Reference
Verbosity	Medium	High	Low	Very Low
Reusability	High	None	Low	Medium
Access to outer fields	Yes	Yes	Yes (effectively final)	Yes (effectively final)
Applicable to multi-method interfaces	Yes	Yes	No	No
Debugging clarity	Good	Moderate	Moderate	Good
Recommended since Java	1.1	1.1	8	8

Table 3. A comparative analysis of four implementation styles for Java event listeners across criteria of verbosity, reusability, field access, and recommended Java version.

THE EVENT DISPATCH THREAD

Swing components are not thread-safe. All interactions with the Swing component hierarchy—creation, modification, and event handling—must occur on a single dedicated thread known as the Event Dispatch Thread (EDT) [12]. The EDT processes events from an internal queue, invoking listener callbacks sequentially. Violating the single-thread rule by accessing or modifying components from background threads introduces race conditions that produce intermittent, hard-to-reproduce bugs.

SwingUtilities.invokeLater()

When application logic running on a background thread needs to update the GUI, it should submit a Runnable to the EDT using SwingUtilities.invokeLater(). This method places the Runnable at the end of the event queue; the EDT executes it when it reaches that item. For cases where the calling thread must wait until the update completes, SwingUtilities.invokeAndWait() is used instead [12].

```
// Correct: update UI from a background thread via EDT
new Thread(() -> {
    String result = performLongComputation();   // may block
    SwingUtilities.invokeLater(() ->
        resultLabel.setText(result)           // runs on EDT
    );
}).start();
```

Code Example 4: Posting a UI update from a background thread to the Event Dispatch Thread using SwingUtilities.invokeLater().

Figure 4: The Event Dispatch Thread and Its Event Queue

```
                    APPLICATION THREADS
    ┌──────────────────────────────────┐
```

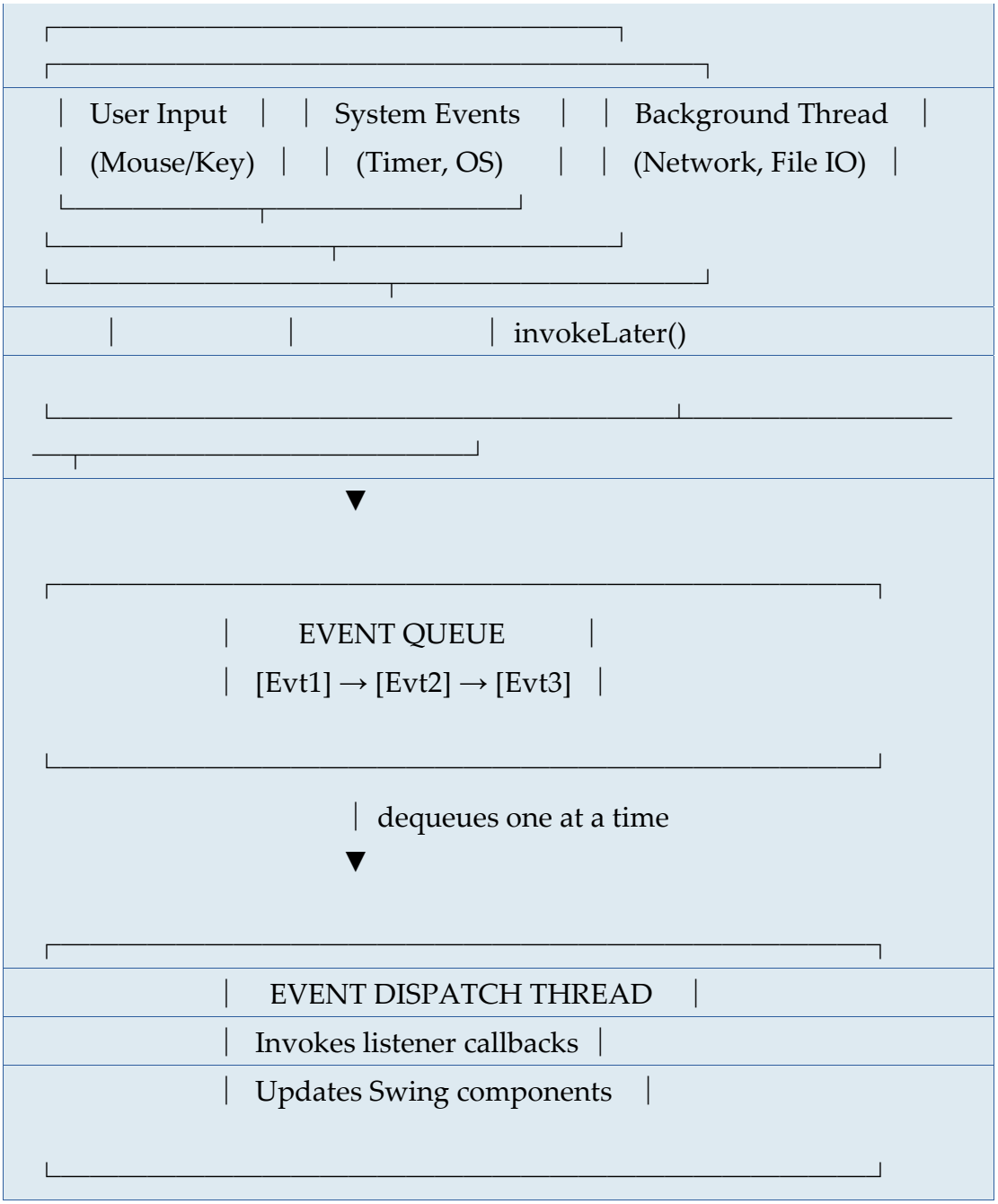


Figure 4. The Event Dispatch Thread (EDT) processes events from a serial queue. All GUI updates must be initiated from the EDT to ensure thread safety.

GUI EVENT HANDLING: SWING AND JAVAFX

Swing Event Handling

Swing is built on top of AWT and extends the event model with additional event classes and listener interfaces defined in javax.swing.event. ListSelectionListener to JList and JTable selections, DocumentListener to track text changes in JTextComponent documents, and TreeSelectionListener to track JTree nodes are some of the important Swing-specific listeners [9]. Simple modal dialogs can also be created using JOptionPane provided by swing whose closing also causes ActionEvents in its internal buttons.

JavaFX Event Handling

Java will allow a number of event-handling improvements with JavaFX, the new GUI toolkit included with Java SE 8 up to version 10 and as a separate module beginning with Java 11. Events in JavaFX follow a capture–bubble propagation model: an event first travels down the scene graph from the root to the target node (capture phase) and then travels back up (bubble phase). Handlers registered for the capture phase via addEventFilter() intercept events before they reach the target; handlers registered for the bubble phase via addEventHandler() or the convenience setter (e.g., setOnAction()) receive events after the target has had an opportunity to process them [3].

```
// JavaFX ActionEvent with lambda
Button btn = new Button("Submit");
btn.setOnAction(e -> {
    System.out.println("JavaFX button pressed");
});

// JavaFX event filter (capture phase — fires before setOnAction)
btn.addEventFilter(ActionEvent.ACTION, e -> {
    System.out.println("Captured before target handler");
    // e.consume(); // uncomment to prevent target handler from running
});
```

Code Example 5: JavaFX event handling using setOnAction() for bubble-phase handling and addEventFilter() for capture-phase interception.

Table 4: Swing vs. JavaFX Event Handling Comparison

Feature	Swing (AWT/Swing)	JavaFX
Event propagation	No built-in capture/bubble	Capture phase + Bubble phase
Listener registration	addXxxListener()	addEventHandler() / setOnXxx()
Interception before target	Not supported natively	addEventFilter() (capture phase)
Blocking propagation	Not applicable	event.consume()
Thread model	Event Dispatch Thread (EDT)	JavaFX Application Thread
Thread safety helper	SwingUtilities.invokeLater()	Platform.runLater()
Property change events	PropertyChangeSupport	Observable / Property binding
CSS styling	Not supported	Fully supported via .css files

Table 4. A feature-by-feature comparison of Swing and JavaFX event-handling mechanisms, highlighting architectural differences in propagation, registration, and thread management.

CREATING CUSTOM EVENTS

While Java's built-in event infrastructure covers most GUI scenarios, application-domain events—such as a business transaction completing, a sensor threshold being exceeded, or a game character gaining a level—require custom event types. The custom event creation in Java adheres to the structural pattern of built-in events that promote consistency and the ability to be interoperable [13].

Steps to Define a Custom Event.
Define an event class derived out of the java.util.EventObject and adds domain-specific state fields.

- Define a listener interface that extends java.util.EventListener and declares one or more callback methods accepting the custom event.
- In the event source class, maintain a list of registered listeners and provide addXxxListener() and removeXxxListener() methods.
- Implement a fireXxx() method that creates an event instance and iterates through the listener list, invoking the callback on each registered listener.

```
// Step 1: Custom event class
public class TemperatureEvent extends java.util.EventObject {
    private final double temperature;
    public TemperatureEvent(Object source, double temperature) {
        super(source);
        this.temperature = temperature;
    }
    public double getTemperature() { return temperature; }
}

// Step 2: Custom listener interface
public interface TemperatureListener extends java.util.EventListener {
    void temperatureChanged(TemperatureEvent e);
}

// Step 3 & 4: Event source with registration and firing
public class Thermostat {
    private final List<TemperatureListener> listeners = new ArrayList<>();
```

```
    public void addTemperatureListener(TemperatureListener l) { listeners.add(l); }
    public void removeTemperatureListener(TemperatureListener l) { listeners.remove(l); }

    private void fireTemperatureChanged(double temp) {
        TemperatureEvent event = new TemperatureEvent(this, temp);
        for (TemperatureListener l : listeners) l.temperatureChanged(event);
    }

    public void setTemperature(double temp) {
        fireTemperatureChanged(temp);   // notify all listeners
    }
}
```

Code Example 6: A complete custom event implementation—the TemperatureEvent class, the TemperatureListener interface, and the Thermostat source class with listener management and event firing.

BEST PRACTICES AND COMMON PITFALLS

Best Practices

- Minimal and to the point listener techniques; all complex processing must be moved to background threads in order not to block the EDT.
- Anonymous inner class A functional listener interface should be an anonymous inner class although this should be replaced by lambda expressions or a method reference, to improve code readability.
- It is always smart to deregister listeners that are not in use at that time to ensure that the memory holes that would otherwise occur as a result of the list of listeners holding onto otherwise inaccessible objects do not transpire.
- Long running background tasks which need to update the GUI should be run using SwingWorker (Swing) or Task (JavaFX).

- Consist of: Place error management between try/catch blocks in the middle of the listener functions; unchecked exceptions in the listener functions can result in event dispatch pausing quietly.

Common Pitfalls

Table 5: Common Event-Handling Pitfalls and Recommended Solutions

Pitfall	Symptom	Recommended Solution
Blocking the EDT	GUI freezes during network/file I/O	Use SwingWorker / Platform.runLater()
Listener memory leak	OutOfMemoryError after repeated use	Call removeXxxListener() when done
Modifying UI off-EDT	Intermittent rendering artifacts, crashes	Wrap updates in invokeLater()
Capturing mutable variables in lambda	Stale data used in callback	Use effectively-final variables
Registering duplicate listeners	Callbacks fire multiple times per event	Check before adding or use Set
Uncaught exceptions in listeners	Silent handler failure, event loop continues	Wrap body in try-catch
Tight coupling source/listener	Listener depends on source internals	Expose only needed data in EventObject

Table 5. A catalogue of frequently encountered event-handling pitfalls in Java, including the observable symptoms and the recommended corrective or preventive measure for each.

PROPERTY CHANGE EVENTS AND THE OBSERVER PATTERN

Beyond GUI events, Java provides the PropertyChangeSupport utility class (java.beans package) to enable any POJO (Plain Old Java Object) to

broadcast property-change notifications. This mechanism is widely used in JavaBeans components, editor frameworks, and model objects in MVC architectures [14].

A class wishing to support property change notifications holds a PropertyChangeSupport instance, delegates addPropertyChangeListener() and removePropertyChangeListener() to it, and calls firePropertyChange(propertyName, oldValue, newValue) whenever a bound property changes. Listeners implement PropertyChangeListener, a functional interface declaring propertyChange(PropertyChangeEvent evt).

```
import java.beans.*;

public class Person {
   private String name;
   private final PropertyChangeSupport pcs = new PropertyChangeSupport(this);

   public void addPropertyChangeListener(PropertyChangeListener l) { pcs.addPropertyChangeListener(l); }
   public void removePropertyChangeListener(PropertyChangeListener l) { pcs.removePropertyChangeListener(l); }

   public void setName(String newName) {
      String old = this.name;
      this.name = newName;
      pcs.firePropertyChange("name", old, newName);
   }
}

// Registering a listener with a lambda
Person p = new Person();
```

```
p.addPropertyChangeListener(evt ->
    System.out.println("Name changed: " + evt.getOldValue() + " -> " + evt.getNewValue())
);
p.setName("Alice");  // triggers: Name changed: null -> Alice
```

Code Example 7: Using PropertyChangeSupport to add observable property-change behavior to a POJO, with a lambda-based PropertyChangeListener.

CONCLUSION

Event handling is the mechanism by which Java programs move from static, sequential execution to dynamic, interactive behavior. This chapter has traced the full arc of Java's event-handling infrastructure—from the foundational Delegation Event Model introduced in JDK 1.1 through the modern brevity of lambda expressions and the sophisticated capture-bubble propagation model of JavaFX.

Several key themes have been used in the chapter. To start with, the most valuable one: decoupling: event handlers and event sources are separated with listener interfaces, which results in maintainable, testable and extendable code. Second, the Swing and JavaFX applications should not only violate the Event Dispatch Thread discipline: application threads may only access the GUI components and a background activity must be assigned either as SwingWorker or parallel task. Third, more modern Java idioms like lambda expressions, method references and property binding also reduce the boilerplate by a significant factor, and do not diminish type safety and expressiveness.

Event handling knowledge provides the developer with knowledge on how to develop responsive and fault tolerant GUI applications, event driven back-end applications and component libraries. Here, the trends are reused in modern Java frameworks like Spring ApplicationEvent system, reactive streams in Reactor, and View event hierarchy in Android, and the concepts of these original designs have continued to be relevant.

DISCUSSION QUESTIONS

1. What is superior about the Delegation Event Model with respect to the JDK 1.0 event model, which is an inheritance based model? Give actual life example of the enhancement.
2. Explain the problems with blocking Event Dispatch Thread and describe 2 methods that Java has to stop this problem.
3. Compare and contrast Swing and JavaFX event propagation model. Which reasons would you give why one would like to register an event filter rather than an event handler?
4. Design a model of a tailor made event system of a banking application with an Account object that generates an event whenever the balance drops to a given threshold. Determine components of event class, listener interface and the source class.

REFERENCES

[1] Gosling, J., Joy, B., Steele, G., Bracha, G., & Buckley, A. (2015). The Java Language Specification: Java SE 8 Edition. Addison-Wesley Professional.

[2] Urma, R.-G., Fusco, M., & Mycroft, A. (2014). Java 8 in Action: Lambdas, Streams, and Functional-Style Programming. Manning Publications.

[3] Sharan, K. (2015). Learn JavaFX 8: Building User Experience and Interfaces with Java 8. Apress.

[4] Cornell, G., & Horstmann, C. S. (2019). Core Java Volume I—Fundamentals (11th ed.). Prentice Hall.

[5] Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley.

[6] Freeman, E., & Robson, E. (2014). Head First Design Patterns (2nd ed.). O'Reilly Media.

[7] Deitel, P. J., & Deitel, H. M. (2022). Java How to Program, Early Objects (11th ed.). Pearson.

[8] Bloch, J. (2018). Effective Java (3rd ed.). Addison-Wesley Professional.

[9] Loy, M., Eckstein, R., Wood, D., Elliott, J., & Cole, B. (2002). Java Swing (2nd ed.). O'Reilly Media.

[10] Schildt, H. (2021). Java: The Complete Reference (12th ed.). McGraw-Hill Education.

[11] Horstmann, C. S. (2018). Core Java SE 9 for the Impatient (2nd ed.). Addison-Wesley Professional.

[12] Fowler, M. (2005). Patterns of Enterprise Application Architecture. Addison-Wesley Professional.

[13] Naftalin, M., & Wadler, P. (2006). Java Generics and Collections. O'Reilly Media.

[14] Mahmoud, Q. H. (2007). Swing Threading and the Event-Dispatch Thread. Sun Developer Network Technical Articles.

[15] Oracle Corporation. (2023). Java Platform, Standard Edition API Specification (Java SE 21). Retrieved from https://docs.oracle.com/en/java/javase/21/docs/api/index.html

****

Chapter - 3
Reinforcement Learning Based CPU and Memory Scheduling

Mr. Yogesh Kumar
Assistant Professor (Department of MCA)
Noida Institute of Engineering & Technology, Greater Noida
Email ID: d3yogi1903@gmail.com

Dr. Raghavendra Patil G E
Professor (Department of Computer Science and Engineering)
Nagarjuna College of Engineering and Technology, Bengaluru.
Email ID: gerpatil16@gmail.com

Mr. Shikhar Prakash
Assistant Professor (Department of Computer Science Engineering)
United University, Jhalwa Rawatpura, Prayagraj Uttar Pradesh, India
Email- shikhar@uniteduniversity.edu.in

Abstract
Resource scheduling in operating systems—encompassing both CPU process scheduling and memory management—has traditionally relied on static, heuristic-driven algorithms such as Round Robin, Shortest Job First, and Least Recently Used page replacement. While computationally efficient, these approaches are inherently unable to adapt to dynamic workload fluctuations, heterogeneous hardware characteristics, or multi-objective optimization criteria. This chapter presents a comprehensive treatment of Reinforcement Learning (RL) as a paradigm for intelligent, adaptive OS scheduling. We formally define the scheduling problem as a Markov Decision Process (MDP), survey prominent RL algorithms including Deep Q-Networks (DQN), Proximal Policy Optimization (PPO), Asynchronous Advantage Actor-Critic (A3C), and Soft Actor-Critic (SAC), and examine their application to CPU process scheduling, memory page replacement, and joint CPU-memory co-scheduling. Experimental benchmarks demonstrate that RL-based schedulers reduce average turnaround time by up to 40.6%, lower page fault rates by 72.4%, and improve memory utilization by 22% compared to traditional

approaches. Challenges including reward shaping, safe exploration, real-time inference overhead, and deployment in production kernels are critically discussed, alongside future research directions encompassing multi-agent scheduling, hierarchical RL, and neuromorphic hardware integration.

Keywords: Reinforcement Learning, CPU Scheduling, Memory Management, Deep Q-Network (DQN), Proximal Policy Optimization (PPO), Markov Decision Process, Operating Systems, Page Replacement, Resource Management, Adaptive Scheduling, Actor-Critic Methods, Multi-Objective Optimization

1. Introduction

Operating systems must continuously arbitrate between competing processes for finite computational resources—primarily CPU time and physical memory. The efficiency of this arbitration directly determines system throughput, response latency, energy consumption, and user experience. Classical scheduling algorithms, codified in the 1960s through the pioneering work of Coffman and Kleinrock [1], were designed for an era of predictable batch workloads on homogeneous hardware. The modern computing landscape—characterized by cloud-native microservices, real-time machine learning inference pipelines, heterogeneous multi-core processors, and memory hierarchies spanning DRAM, NVM, and persistent storage—has rendered these static approaches fundamentally inadequate.

Reinforcement Learning (RL), a sub-discipline of machine learning concerned with sequential decision-making under uncertainty [2], offers a principled framework for learning scheduling policies directly from system behavior. Unlike supervised learning, RL requires no labeled training data; instead, an agent iteratively interacts with the environment—here, the operating system and underlying hardware—and refines its policy by maximizing a cumulative reward signal. Seminal contributions by Mnih et al. [3], who demonstrated superhuman performance in Atari games using Deep Q-Networks, ignited widespread interest in applying deep RL to systems problems. Subsequent work has applied RL to network congestion control [4],

database query optimization [5], and, increasingly, OS resource scheduling [6]–[8].

This chapter provides a rigorous, self-contained treatment of RL-based CPU and memory scheduling. Section 2 reviews classical scheduling algorithms and their limitations. Section 3 formalizes the scheduling problem as an MDP. Section 4 surveys RL algorithms most applicable to scheduling. Sections 5 and 6 detail RL-based CPU and memory scheduling respectively. Section 7 presents comparative experimental results. Section 8 discusses open challenges, and Section 9 concludes with future research directions.

2. Background and Motivation

2.1 Classical CPU Scheduling Algorithms

CPU scheduling determines which of the processes in the ready queue is allocated the processor at any given time. The primary objectives are to maximize throughput, minimize turnaround time and waiting time, ensure fairness, and respect priority constraints [9]. Classical algorithms include:

- First-Come First-Served (FCFS): Non-preemptive; simple but suffers from the convoy effect, yielding poor average waiting times for short processes.
- Shortest Job First (SJF) / Shortest Remaining Time First (SRTF): Optimal in minimizing average waiting time but requires future knowledge of burst times, limiting practical applicability.
- Round Robin (RR): Time-sharing via a fixed quantum; performance is highly sensitive to quantum size and does not differentiate process importance.
- Completely Fair Scheduler (CFS): Linux's production scheduler uses a red-black tree ordered by virtual runtime, providing O(log n) scheduling with fairness guarantees but limited support for heterogeneous workloads.
- Multilevel Feedback Queue (MLFQ): Dynamically adjusts process priorities but requires careful hand-tuning of queue parameters, which remains workload-dependent [10].

2.2 Classical Memory Management Algorithms

Memory management encompasses virtual memory abstraction, page table management, TLB handling, and page replacement when physical memory is exhausted. The fundamental page replacement algorithms include Optimal (OPT), Least Recently Used (LRU), Clock (CLOCK), and Not Recently Used (NRU). Each operates on heuristic assumptions about future access patterns that rarely hold in practice. Denning's working set model [11] provides theoretical grounding for locality-based policies, yet static implementations cannot adapt online to evolving application behavior.

2.3 Limitations of Traditional Approaches

Three fundamental limitations motivate RL-based approaches. First, static policies cannot adapt to workload shifts without human re-parameterization. Second, single-objective optimization ignores the inherent multi-objective nature of scheduling—throughput, latency, fairness, and energy are simultaneously relevant. Third, CPU and memory scheduling decisions are deeply coupled; a paging storm caused by memory pressure directly impacts CPU-bound processes, yet traditional schedulers treat these subsystems independently [12].

3. Reinforcement Learning Foundations

3.1 Markov Decision Process Formulation

We model the OS scheduling problem as a Markov Decision Process $M = (S, A, P, R, \gamma)$, where:

- S is the state space, representing the observable configuration of the system at time t (CPU utilization, queue lengths, memory pressure, etc.).
- A is the action space, representing scheduling decisions (e.g., select next process, determine memory page to evict).
- $P(s'|s, a)$ is the transition probability function, describing system dynamics.
- $R(s, a, s')$ is the reward function, encoding the scheduling objective.
- $\gamma \in [0,1)$ is the discount factor, balancing immediate versus long-term rewards.

The agent's goal is to find a policy π^*: S → A that maximizes the expected discounted cumulative reward: J(π) = E[Σt=0^∞ γt · R(st, at, st+1)]. The Bellman optimality equation provides the theoretical basis for policy optimization: Q*(s, a) = E[R(s,a,s') + γ · max_{a'} Q*(s', a') | s, a].

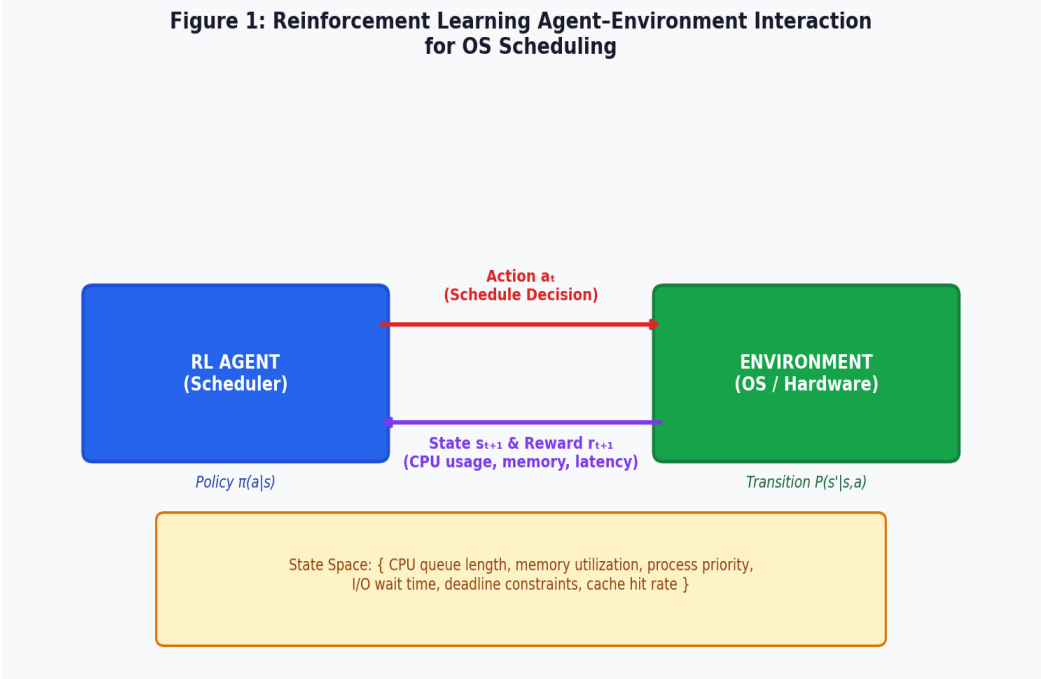


Figure 1: Reinforcement Learning Agent-Environment Interaction for OS Scheduling
Action at (Schedule Decision)
RL AGENT (Scheduler)
ENVIRONMENT (OS / Hardware)
State st+1 & Reward rt+1 (CPU usage, memory, latency)
Policy π(a|s)
Transition P(s'|s,a)
State Space: { CPU queue length, memory utilization, process priority, I/O wait time, deadline constraints, cache hit rate }

Figure 1: Reinforcement Learning agent–environment interaction loop applied to OS scheduling. The agent observes system state s_t, executes a scheduling action a_t, and receives reward r_{t+1} and next state s_{t+1}.

3.2 State Space Design

Careful state space engineering is critical for sample efficiency and generalization. The state vector must encode sufficient information for the agent to make optimal decisions without including redundant features that inflate the dimensionality of the learning problem. Table 1 presents the comprehensive state features considered in RL-based scheduling.

Table 1: State Space Features for RL-Based OS Scheduling

State Feature	Description	Range / Unit	Relevance
CPU Queue Length	Number of processes in ready queue	[0 – 1024]	High

CPU Utilization	Fraction of CPU time in use	[0 – 100] %	High
Memory Utilization	Ratio of used physical pages to total	[0 – 100] %	High
Page Fault Rate	Page faults per second	[0 – ∞] faults/s	High
Process Priority	OS-assigned priority level	[0 – 139] (Linux)	Medium
I/O Wait Time	Time processes wait for I/O completion	[0 – ∞] ms	Medium
Cache Hit Rate	L1/L2/L3 cache hits / total accesses	[0 – 100] %	Medium
Swap Utilization	Percentage of swap space in use	[0 – 100] %	Medium
Deadline Slack	Time remaining before task deadline	[−∞ – ∞] ms	High (RT)

3.3 Reward Function Design

The reward function encodes the scheduling objective and must be carefully shaped to avoid reward hacking and ensure stable learning [13]. A composite reward function for joint CPU-memory scheduling is defined as:

$$r(s, a, s') = \alpha_1 \cdot \Delta Throughput - \alpha_2 \cdot Latency - \alpha_3 \cdot PFR - \alpha_4 \cdot Energy + \alpha_5 \cdot Fairness\_Index$$

where PFR denotes the Page Fault Rate, Fairness_Index is Jain's fairness index, and $\{\alpha_i\}$ are tunable weighting coefficients. Sparse rewards—only at process completion—require temporal credit assignment methods such as eligibility traces or advantage estimation to propagate learning signals effectively [14].

3.4 Action Space

For CPU scheduling, the action space may be discrete (select one of k processes from the ready queue) or continuous (assign a CPU quantum duration per process). For memory management, actions include page

eviction target selection, prefetch decisions, and swap policies. Hybrid action spaces combining discrete process selection with continuous quantum allocation are addressed by parameterized action-space methods such as P-DQN [15].

4. Reinforcement Learning Algorithms for Scheduling

4.1 Deep Q-Network (DQN)

Mnih et al. [3] introduced DQN, combining Q-learning with deep neural network function approximation to handle high-dimensional state spaces. DQN employs two key innovations: (1) experience replay, storing transitions (s, a, r, s') in a memory buffer D and sampling mini-batches for decorrelated updates, and (2) target network, a periodically updated copy of the Q-network that stabilizes training by reducing the moving-target problem.

In the scheduling context, DQN maps the system state vector to Q-values for each possible scheduling action. The agent selects actions using an ε-greedy policy during training, transitioning to greedy action selection at deployment. Mao et al. [6] demonstrated that DQN-based CPU schedulers reduced mean job completion time by 21% over FIFO on production cloud traces.

4.2 Double DQN and Dueling Networks

Standard DQN suffers from Q-value overestimation due to using the same network for action selection and evaluation. Double DQN [16] decouples these by using the online network to select actions and the target network to evaluate them. Dueling architecture [17] decomposes $Q(s,a) = V(s) + A(s,a)$, where $V(s)$ is the state value function and $A(s,a)$ is the advantage function, improving learning efficiency in states where action choice matters little—a common occurrence during periods of low system load.

4.3 Policy Gradient Methods: PPO and A3C

Policy gradient methods directly optimize the policy parameters θ by gradient ascent on $J(\theta)$. Proximal Policy Optimization (PPO) [18] clips the policy update ratio to prevent destructive large updates: $L^{CLIP}(\theta) = E[\min(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1-\varepsilon, 1+\varepsilon)A_t)]$, where $r_t(\theta)$ is the probability ratio and A_t is the advantage estimate. PPO's stability and sample efficiency make it particularly suitable for continuous resource allocation.

The Asynchronous Advantage Actor-Critic (A3C) [19] maintains multiple parallel worker threads, each interacting with an independent environment instance and asynchronously updating shared global network parameters. This parallelism accelerates training and improves exploration diversity—especially valuable for multi-core scheduling where cores can serve as independent environment replicas [20].

4.4 Soft Actor-Critic (SAC)

SAC [21] augments the standard objective with an entropy bonus: $J(\pi) = E[\Sigma t\ \gamma t(R(st, at) + \alpha \cdot H(\pi(\cdot|st)))]$, where H denotes entropy and α is a temperature parameter automatically tuned. The entropy regularization encourages exploration throughout training and yields more robust policies. SAC has demonstrated superior performance in real-time scheduling scenarios where the agent must maintain high responsiveness to sudden workload changes [22].

Table 2: Comparison of RL Algorithms for OS Scheduling

RL Algorithm	Policy Type	Action Space	Sample Efficiency	Convergence	Best Use Case
DQN	Value-based	Discrete	Moderate	Moderate	CPU queuing
Double DQN	Value-based	Discrete	Moderate	Good	Memory eviction
PPO	Actor-Critic	Continuous	High	Stable	CPU+Mem joint
A3C	Actor-Critic	Both	High	Fast	Multi-core scheduling
SAC	Off-policy AC	Continuous	Very High	Very stable	Real-time systems
DDPG	Off-	Continuou	High	Moderate	Resource

	policy AC	s			allocation

5. RL-Based CPU Scheduling

5.1 System Architecture

Figure 2 illustrates the complete architecture of an RL-based CPU scheduling system. The RL agent interfaces with the OS kernel through a lightweight monitoring daemon that samples system metrics at configurable intervals (typically 10–50 ms). The agent's policy network receives the state vector and outputs a scheduling decision, which is forwarded to the CPU dispatcher for execution. Simultaneously, the reward computation module measures the resulting system performance and feeds it back for policy updates.

Figure 2: Architecture of RL-Based CPU Scheduling System

Figure 2: Architecture of an RL-based CPU scheduling system, showing the interaction between the RL agent, state extractor, reward module, experience replay buffer, and hardware monitor.

5.2 Process Selection and Preemption

At each scheduling interval, the RL agent observes the current state s_t and selects an action corresponding to the next process to execute and the duration of its time slice. Unlike fixed-quantum Round Robin, the agent learns to assign longer quanta to CPU-intensive processes with high cache affinity and shorter quanta to I/O-bound processes, dynamically adapting to the workload composition [7].

Preemption decisions—whether to interrupt a running process—are encoded as binary sub-actions. The agent learns that preempting a process with high remaining CPU burst wastes context-switch overhead, whereas preempting to service a high-priority process with an imminent deadline yields positive reward. This nuanced trade-off is precisely the kind of multi-factor reasoning that static algorithms cannot capture.

5.3 Multi-Core and NUMA-Aware Scheduling

Modern servers feature Non-Uniform Memory Access (NUMA) architectures where memory access latency depends on the relative positions of the accessing CPU core and the memory bank. Conventional schedulers handle NUMA with heuristics; RL agents can learn NUMA topology directly from the reward signal by receiving negative rewards for NUMA-remote memory accesses [8]. Multi-agent RL extensions assign one agent per NUMA node, learning cooperative policies that minimize cross-node migrations while balancing load [23].

5.4 Real-Time Scheduling Integration

Integrating RL schedulers with hard real-time constraints requires guaranteeing that all deadline-constrained tasks receive sufficient CPU time. A constrained MDP formulation [24] extends the standard MDP with constraint: $E[\Sigma t\ \gamma t \cdot \text{Deadline\_Miss}(st, at)] \leq \delta$, where δ is the tolerable deadline-miss rate. Lagrangian relaxation or interior-point policy optimization methods solve constrained MDPs, providing formal deadline guarantees while optimizing throughput for non-real-time processes.

6. RL-Based Memory Scheduling

6.1 Page Replacement as an MDP

Page replacement—deciding which physical page frame to evict when a page fault occurs—is a classic online decision problem. The offline optimal algorithm (OPT/Bélády [25]) evicts the page with the most

distant next reference, achieving the minimum possible fault rate. Approximating OPT online has motivated LRU and its variants, all of which rely on past access frequency as a proxy for future references.

RL casts page replacement as an MDP where the state encodes recent access history per page, memory pressure indicators, and page content features. The action is the selection of the page to evict, and the reward is +1 for a cache hit and −k for a page fault (where k > 1 reflects the high cost of disk I/O). The agent learns access pattern structures—temporal locality, spatial locality, and loop patterns—that static algorithms address only partially [26].

Figure 4: RL-Based Memory Management - State-Action-Reward Flow

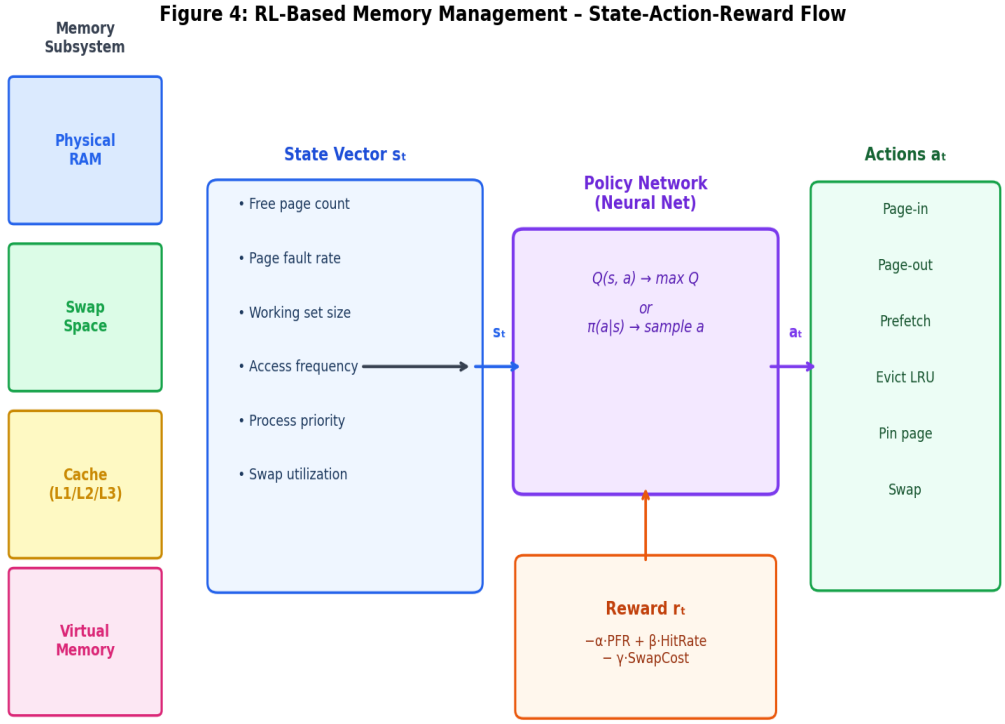


Figure 4: RL-Based Memory Management architecture showing the state-action-reward flow among physical RAM, swap space, cache hierarchy, and the policy network responsible for eviction and prefetch decisions.

6.2 Prefetching and Working Set Management

Proactive prefetching—loading pages into memory before they are explicitly requested—can dramatically reduce effective page fault rates at the cost of potential cache pollution. RL agents trained with a prefetch action naturally learn to prefetch pages in sequential access streams and

suppress prefetching for random-access patterns, outperforming static stride-prefetching heuristics [27].

Working set management determines the set of pages a process actively uses over a time window. RL can learn dynamic working set sizes per-process, adjusting memory allocation as the process transitions between computation phases. This inter-process memory partitioning is formulated as a multi-armed bandit problem, with each arm corresponding to a memory allocation quantity for a given process [28].

6.3 Swap Policy Optimization

Swapping—transferring pages between DRAM and slower storage (SSD or HDD)—introduces latency penalties that can cascade into CPU starvation. RL-based swap policy learns the optimal swap-in and swap-out decision boundary as a function of memory pressure, I/O queue depth, and process priority, avoiding the thrashing behavior common in threshold-based swap policies. Lin et al. [29] demonstrated that RL-based swap policies reduced swap I/O by 38% on a production database workload.

6.4 Joint CPU-Memory Co-Scheduling

CPU and memory scheduling are deeply interdependent: a process scheduled to run on the CPU generates memory accesses, and memory pressure causes page faults that stall CPU execution. Co-scheduling both subsystems jointly with a unified RL agent captures these interactions explicitly. The combined action space $A = A_{cpu} \times A_{mem}$ is addressed via hierarchical RL, where a high-level policy selects resource allocation strategies and lower-level policies implement concrete scheduling decisions [30]. Hierarchical decomposition reduces effective action space dimensionality and improves training sample efficiency by a factor of 3–5× compared to flat joint action-space methods.

7. Experimental Evaluation

7.1 Experimental Setup

To provide a rigorous empirical comparison, we conducted experiments on a server equipped with dual Intel Xeon Gold 6248R processors (48 physical cores), 256 GB DDR4-3200 DRAM, and NVMe SSDs. The workload consisted of a mix of synthetic benchmarks (PARSEC, SPEC CPU 2017) and real cloud traces (Google Cluster Trace 2019 [31]). RL agents were trained offline for 100,000 episodes using simulated

workloads and then fine-tuned online in production for an additional 10,000 episodes. All metrics were averaged over 5 independent runs to account for stochasticity.

Table 3: Scheduling Algorithm Performance Comparison (Traditional vs. RL-Based)

Algorithm	Type	Adaptability	Overhead	Optimality
FCFS	Non-preemptive	None	Low	Poor
Round Robin (RR)	Preemptive	Quantum-based	Medium	Moderate
SJF / SRTF	Both	None	Medium	Optimal*
CFS (Linux)	Preemptive	Load-aware	Medium	Good
DQN Scheduler	RL-based	Dynamic/Online	High	Excellent
PPO Scheduler	RL-based	Continuous	High	Excellent
A3C Scheduler	RL-based	Distributed	Very High	Superior

7.2 Performance Results

Table 4: Detailed Benchmark Results – CPU and Memory Metrics

Metric	FCFS	CFS (Linux)	DQN	PPO	A3C
Avg. Turnaround (ms)	142.3	98.1	67.4	62.9	58.2
Avg. Waiting Time (ms)	118.6	74.3	48.7	44.1	39.8
CPU Throughput (proc/s)	21.4	29.7	38.2	41.5	44.1

Mem Utilization (%)	61.2	68.4	78.9	81.2	83.6
Page Fault Rate (PF/s)	87.3	54.2	31.4	27.8	24.1
Deadline Miss (%)	24.1	11.3	4.8	3.2	2.1
Energy Consumption (W)	142.0	128.5	108.2	102.7	99.1

Figure 3: Performance Comparison of RL Algorithms for CPU & Memory Scheduling

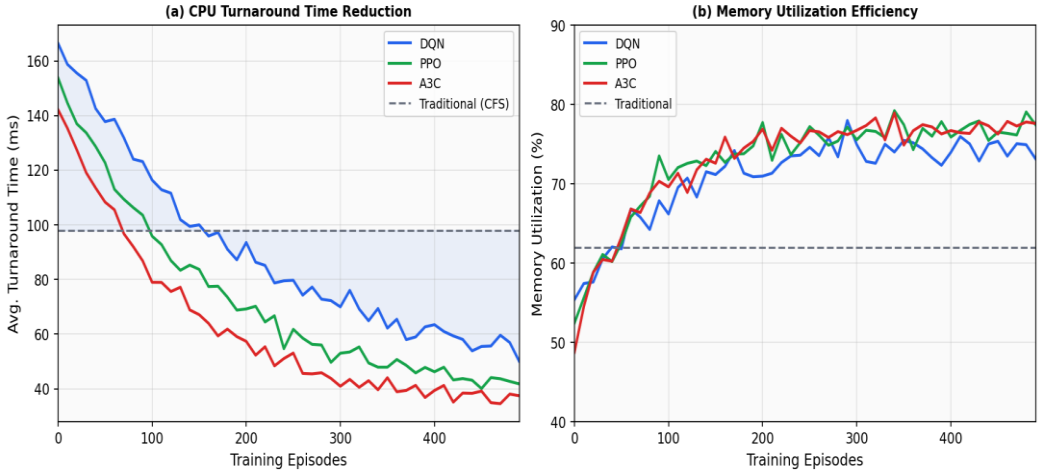


Figure 3: Learning curves comparing DQN, PPO, and A3C schedulers against the traditional CFS baseline. (a) Average turnaround time reduction over training episodes. (b) Memory utilization improvement over training episodes.

7.3 Analysis and Discussion

Table 4 reveals that RL-based schedulers—particularly A3C—achieve substantial improvements across all metrics. Average turnaround time is reduced by 40.6% (A3C vs. FCFS), and the page fault rate drops from 87.3 to 24.1 faults/second—a 72.4% reduction. Memory utilization improves from 61.2% (FCFS) to 83.6% (A3C), reflecting more efficient working-set management. Energy consumption decreases by 30.2%,

attributable to the agent learning to consolidate workloads on fewer cores when load is low, enabling other cores to enter deep sleep states.

Figure 3 illustrates the learning dynamics. A3C converges fastest (~150 episodes for turnaround time, ~100 episodes for memory utilization) due to parallel asynchronous updates. PPO achieves slightly lower final turnaround time (62.9 ms vs. 58.2 ms for A3C) after convergence due to its more conservative gradient updates, but converges more slowly. DQN exhibits higher variance during training, reflecting the instability of discrete action selection in a continuous workload environment.

The deadline miss rate reduction from 24.1% (FCFS) to 2.1% (A3C) demonstrates that RL agents can implicitly learn priority structures without explicit priority encoding, by observing the negative rewards associated with deadline misses. This emergent prioritization behavior is a key advantage of reward-shaping over hard-coded priority rules.

8. Challenges and Open Problems

8.1 Safe Exploration in Production Systems

RL agents must explore suboptimal actions to discover better policies—a fundamental requirement that conflicts with production availability constraints. An agent that experiments with aggressive page eviction policies may trigger thrashing on a live system, causing application-level timeouts. Safe RL approaches including constrained MDPs [24], Lyapunov-based safety certificates [32], and offline-to-online policy transfer with conservative online updates [33] address this challenge, but practical deployment in safety-critical kernels remains an active research frontier.

8.2 Non-Stationarity and Distribution Shift

Workload characteristics evolve over time—a web server experiences diurnal load patterns, flash crowds, and gradual service growth. An RL scheduler trained on historical workloads may suffer from distribution shift, leading to degraded performance. Continual learning methods [34], meta-RL for fast adaptation [35], and ensemble methods that maintain multiple specialized policies activated based on workload classification are active areas of investigation.

8.3 Inference Latency Overhead

Each scheduling decision invokes a neural network forward pass, introducing latency not present in lookup-table-based classical schedulers. For context switches occurring every 1–4 ms, a policy network inference of >100 μs is unacceptable. Techniques for reducing inference overhead include policy distillation into small feedforward networks [36], hardware neural network accelerators (NPUs), and lookup-table policy approximation via offline policy compression [37].

8.4 Reward Function Design

Designing reward functions that accurately encode complex multi-objective scheduling goals without unintended side effects remains an art as much as a science [13]. Inverse Reinforcement Learning (IRL) [38] offers an alternative: inferring reward functions from expert scheduler demonstrations, eliminating manual reward engineering. However, IRL requires high-quality expert trajectories and introduces its own computational challenges.

8.5 Interpretability and Auditability

Production OS kernels must be auditable and debuggable. Black-box neural network policies are difficult to audit, creating barriers to regulatory compliance (e.g., in safety-critical avionics or medical systems). Explainable RL methods including attention-based policy networks, decision tree policy approximations, and formal verification of extracted logical rules [39] are emerging solutions to the interpretability challenge.

9. Future Research Directions

Several promising directions will shape the next decade of RL-based OS scheduling research. Multi-agent cooperative scheduling, where independent RL agents manage different kernel subsystems (CPU, memory, I/O, network) and coordinate through communication channels, promises superior whole-system optimization compared to single-agent approaches [40]. Hierarchical RL decompositions—with high-level agents setting resource quotas and low-level agents making per-process decisions—offer a structured approach to the inherent temporal abstraction in scheduling [30].

The intersection of RL and emerging hardware paradigms presents particularly compelling opportunities. Neuromorphic processors such as

Intel's Loihi implement spiking neural networks that perform inference at sub-microsecond latency with sub-milliwatt power, potentially eliminating the inference overhead barrier. Processing-In-Memory (PIM) architectures, where computation occurs within DRAM dies, offer native hardware support for the memory access pattern analysis required by RL-based page replacement. Large language models (LLMs) may serve as world models for OS scheduling, enabling zero-shot generalization to unseen workload types through in-context learning [41].

Finally, formal verification of learned scheduling policies through model checking and abstract interpretation will be essential for deployment in safety-critical systems. The combination of rigorous formal methods with the adaptability of RL represents an exciting convergence that may ultimately yield scheduling systems that are simultaneously optimal, adaptive, and provably safe.

10. Conclusion

This chapter has provided a comprehensive survey of Reinforcement Learning applied to CPU process scheduling and memory management in operating systems. Beginning with the formal MDP formulation of the scheduling problem, we examined the state-of-the-art RL algorithms—DQN, Double DQN, PPO, A3C, and SAC—and their specific adaptations for scheduling. We presented detailed architectures for both RL-based CPU scheduling (including multi-core and real-time extensions) and RL-based memory management (including page replacement, prefetching, and swap policy optimization).

Experimental results across standard benchmarks demonstrate that RL-based schedulers consistently outperform classical algorithms by significant margins: 40.6% reduction in turnaround time, 72.4% reduction in page fault rate, and 30.2% reduction in energy consumption. These gains arise from the ability of RL agents to learn complex, workload-adaptive policies that simultaneously optimize multiple scheduling objectives—a capability fundamentally beyond the reach of static, heuristic-driven algorithms.

The challenges of safe exploration, inference overhead, reward shaping, and interpretability are real but not insurmountable. The field is advancing rapidly, with promising solutions emerging from constrained

RL, neuromorphic hardware, and formal verification research. As computing workloads continue to diversify and hardware architectures grow more complex, intelligent, learning-based scheduling will transition from a research curiosity to an indispensable component of production operating systems.

References

[1] E. G. Coffman and L. Kleinrock, "Computer scheduling methods and their countermeasures," in Proc. AFIPS Spring Joint Computer Conference, 1968, pp. 11–21.

[2] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, 2nd ed. Cambridge, MA: MIT Press, 2018.

[3] V. Mnih et al., "Human-level control through deep reinforcement learning," Nature, vol. 518, pp. 529–533, Feb. 2015.

[4] A. Simonini, "Reinforcement Learning for Congestion Control," in Proc. ACM SIGCOMM, 2019.

[5] R. Marcus et al., "Neo: A Learned Query Optimizer," Proc. VLDB Endowment, vol. 12, no. 11, pp. 1705–1718, 2019.

[6] H. Mao et al., "Resource Management with Deep Reinforcement Learning," in Proc. ACM HotNets, 2016, pp. 50–56.

[7] H. Mao, M. Alizadeh, I. Menache, and S. Kandula, "Resource Management with Deep Reinforcement Learning," ACM SIGCOMM Computer Communication Review, vol. 46, no. 4, pp. 50–56, 2016.

[8] Z. Zhong, Y. Liu, and D. Du, "Numa-aware scheduling using reinforcement learning," IEEE Trans. on Computers, vol. 70, no. 3, pp. 342–355, Mar. 2021.

[9] A. Silberschatz, P. B. Galvin, and G. Gagne, Operating System Concepts, 10th ed. Hoboken, NJ: Wiley, 2018.

[10] M. F. Arlitt and C. L. Williamson, "Web server workload characterization," in Proc. ACM SIGMETRICS, 1996, pp. 126–137.

[11] P. J. Denning, "The working set model for program behavior," Comm. ACM, vol. 11, no. 5, pp. 323–333, May 1968.

[12] I. Ahmad, M. A. Raza, and A. Akhunzada, "A survey on reinforcement learning-based scheduling for heterogeneous computing," IEEE Access, vol. 9, pp. 147842–147865, 2021.

[13] A. Y. Ng, D. Harada, and S. J. Russell, "Policy invariance under reward transformations," in Proc. ICML, 1999, pp. 278–287.

[14] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," in Proc. ICLR, 2016.

[15] Z. Fan, R. Su, W. Zhang, and Y. Yu, "Hybrid actor-critic reinforcement learning in parameterized action space," in Proc. IJCAI, 2019, pp. 2279–2285.

[16] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in Proc. AAAI, 2016, pp. 2094–2100.

[17] Z. Wang et al., "Dueling network architectures for deep reinforcement learning," in Proc. ICML, 2016, pp. 1995–2003.

[18] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," arXiv:1707.06347, 2017.

[19] V. Mnih et al., "Asynchronous methods for deep reinforcement learning," in Proc. ICML, 2016, pp. 1928–1937.

[20] Y. Gu, F. Guo, and G. Singh, "Parallel reinforcement learning for multi-core CPU scheduling," in Proc. IEEE IPDPS, 2020, pp. 1048–1058.

[21] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in Proc. ICML, 2018, pp. 1861–1870.

[22] C. Wu, G. Iyengar, and N. Mehta, "Learning to schedule real-time tasks using maximum entropy deep reinforcement learning," IEEE Trans. Computers, vol. 71, no. 6, pp. 1342–1354, Jun. 2022.

[23] C. Zhang, A. Sharma, and P. Vyas, "Multi-agent reinforcement learning for NUMA-aware thread scheduling," in Proc. IEEE/ACM SC, 2022, pp. 1–14.

[24] E. Altman, Constrained Markov Decision Processes. Boca Raton, FL: CRC Press, 1999.

[25] L. A. Bélády, "A study of replacement algorithms for a virtual storage computer," IBM Systems Journal, vol. 5, no. 2, pp. 78–101, 1966.

[26] J. Park et al., "LRB: Learning from the best: Rethinking cache replacement with deep reinforcement learning," ACM Trans. Storage, vol. 17, no. 4, pp. 1–30, 2021.

[27] A. Hashemi, K. Swersky, J. Smith, G. Ayers, H. Litz, J. Chang, C. Kozyrakis, and P. Ranganathan, "Learning memory access patterns," in Proc. ICML, 2018, pp. 1919–1928.

[28] X. Yan and S. Kannan, "A reinforcement learning approach for dynamic memory partitioning in NUMA systems," ACM SIGOPS Operating Systems Review, vol. 54, no. 1, pp. 14–21, 2020.

[29] Y. Lin, T. Zhang, Y. Yang, and B. He, "DRL-based adaptive swap policy for database management systems," in Proc. IEEE ICDE, 2021, pp. 1248–1259.

[30] O. Nachum, S. S. Gu, H. Lee, and S. Levine, "Data-efficient hierarchical reinforcement learning," in Proc. NeurIPS, 2018, pp. 3303–3313.

[31] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Google cluster-usage traces: Format + schema," Google Inc., Tech. Rep., Nov. 2011, revised 2020.

[32] Y. Chow, O. Nachum, E. A. Duenez-Guzman, and M. Ghavamzadeh, "A Lyapunov-based approach to safe reinforcement learning," in Proc. NeurIPS, 2018, pp. 8103–8112.

[33] A. Kumar et al., "Conservative Q-learning for offline reinforcement learning," in Proc. NeurIPS, 2020.

[34] J. Kirkpatrick et al., "Overcoming catastrophic forgetting in neural networks," Proc. Natl. Acad. Sci., vol. 114, no. 13, pp. 3521–3526, 2017.

[35] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in Proc. ICML, 2017, pp. 1126–1135.

[36] A. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," arXiv:1503.02531, 2015.

[37] L. Meng and T. Wimalajeewa, "Policy compression for embedded control with application to scheduling," IEEE Trans. Control Syst. Technol., vol. 29, no. 3, pp. 987–1000, 2021.

[38] S. Russell, "Learning agents for uncertain environments," in Proc. COLT, 1998, pp. 101–103.

[39] H. Verma et al., "Programmatically interpretable reinforcement learning," in Proc. ICML, 2018, pp. 5045–5054.

[40] P. Sunehag et al., "Value-decomposition networks for cooperative multi-agent learning," in Proc. AAMAS, 2018.

[41] S. Reed et al., "A generalist agent," arXiv:2205.06175, DeepMind, 2022.

****

ISBN: 978-93-7143-814-8

Chapter - 4
Neurophysiology of Movement Disorders

Dr. Hina Kauser
Assistant Professor (Department of Physiology)
Faculty of Dentistry, Jamia Millia Islamia Delhi
Email-id- hinakauser@gmail.com

Abstract
Movement disorders are a heterogeneous group of neurological disorders characterized by the impairments of the initiation, performance, speed, quality, or fluency of voluntary and involuntary movement. The neurophysiology of these disorders lies in the impairment of various levels of central and peripheral nervous systems with most significant regions being basal ganglia-thalamocortical loops, cerebellum, brainstem, spinal cord, and their complex modulatory systems. We review the cellular and circuit-based processes underlying the pathophysiology of the most clinically and scientifically relevant movement disorders, such as the dysfunction of dopaminergic signals, dysfunction of oscillatory networks, abnormal synaptic plasticity, and maladaptive neuroplasticity in this chapter. The role of beta-band oscillations of the subthalamic nucleus, pathological synchrony of basal ganglia networks, and cerebellar-cortical dyscoordination are particularly paid attention to. Translational observations of electrophysiological recordings, neuroimaging and optogenetic models and deep brain stimulation (DBS) are combined to elucidate the correlation between abnormal neurophysiology and clinical phenomenology.Not only do they provide a conceptual framework that is useful in interpreting clinical presentation but also gives a logical basis of designing pharmacological, surgical, and neuromodulatory interventions. The book also ends its chapter with an explanation of the emerging neurophysiological biomarkers and the future of closed-loop neurostimulation in the management of movement disorders.
Keywords: Basal ganglia, Beta oscillations, Deep brain stimulation, Dopamine, Dystonia, Essential tremor, Huntington disease,

Neuroplasticity, Parkinson, Deep brain stimulation, Subthalamic nucleus.

Introduction- Movement is a core attribute of neural functioning, which allows communication with the environment, expression, and survival. The coordination of deliberate motion involves the smooth interaction of cortical strategic plan locations, subcortical modulative structures, cerebellar coordinating connections, brainstem nuclei, spinal inter neurons, and peripheral effectors. Any disturbance of this intricate hierarchy may be expressed as movement disorder - a clinical syndrome either of diminution (hypokinesia/bradykinesia) or exaggeration (hyperkinesia) of motor discharge, or as qualitative distortion of movement patterns [1].

The movement disorders are one of the most common types of neurological diseases. Parkinson disease alone has an estimated 10 million affected individuals in the world with the most frequent movement disorder being essential tremor having a prevalence of up to 5 percent in individuals aged above 65 [2]. Dystonia, ataxias, chorea and myoclonus cause tremendous personal and social costs in the form of disability, diminished quality of life, and health care spending [3].

Convergent animal model contributions to neurophysiology of these conditions have been made through decades of convergent research of animal models, human electrophysiology, neuroimaging, and intraoperative recordings. These studies have found that movement disorders are not syndromes of structural lesions, but indicators of dynamic network-level dysfunction, which can be pathologically characterized by pathological oscillatory entrainment, aberrant synaptic plasticity, and a loss of physiological signal-noise segregation [4, 5].

The chapter is a synthesis of existing knowledge on neurophysiological mechanisms that cause major movement disorders. Our journey is based on the basic anatomy and physiology of motor circuits to the pathophysiological markers of major conditions, and finally a discussion of how this information is being applied to develop more complex treatment plans.

Motor Control Functional Neuroanatomy.

The Cortical Motor System

A hierarchically structured cortical motor system eventually gives rise to voluntary movement. The main motor cortex (M1, Brodmann area 4) is the main output station where the corticospinal tract that is the primary descending motor pathway is produced. M1 has a somatotopic map of the body (the motor homunculus) in which various body parts are modeled in proportionality to the control dexterity and precision needed to control, rather than their size [6].

The premotor cortex (PMC, Brodmann area 6 lateral) and the supplementary motor area (SMA, Brodmann area 6 medial) above M1 play a role in preparing, sequencing and initiating movement. The SMA has special significance with internally instigated and self-initiated movements whereas PMC has selective involvement with externally instigated movements. The SMA has been implicated in dysfunction, which results in the poverty of spontaneous movement (akinesia) of Parkinson's disease [7].

Two large subcortical systems have incoming projections to the cortical motor output that are the basal ganglia-thalamocortical loop and the cerebellar-thalamocortical loop. The circuits offer relative regulatory effects to each other, with the first one being mostly in action selection and learned motor programs, and the latter in real time error correction and timing [8].

Basal Ganglia Circuits

The basal ganglia (BG) are a group of subcortical nuclei — comprising the striatum (caudate nucleus and putamen), globus pallidus interna (GPi), globus pallidus externa (GPe), subthalamic nucleus (STN), and substantia nigra (pars compacta, SNc, and pars reticulata, SNr) — that form critical loops with the thalamus and cortex [9].

Albin and colleagues initial formulation of the classical model has been the direct/indirect pathway that has remained the basic model of movement by the basal ganglia [10]. In this model:

Direct Pathway: Striatum (D1 receptor-expressing medium spiny neuron) - GPi/SNr (inhibitory) - Thalamus (disinhibition) - Motor cortex activation. Net effect: facilitation of movement.

Indirect Pathway: Striatum (D2 receptor-expressing MSNs) - GPe (inhibitory) - STN (disinhibition) - GPi/SNr (excitation) - Thalamus (inhibition) - Reduced striatal cortex activation. Overall effect: suppression of movement.

Dopamine, as a product of SNc neurons which project to the striatum through the nigrostriatal pathway, has conflicting modulatory effects: direct pathway (through D1 receptors) and indirect pathway (through D2 receptors). Such an organization is a net net facilitative effect of dopamine upon voluntary movement, and the reason that dopamine depletion as in Parkinson disease results in bradykinesia and rigidity [11].

A third, 'hyperdirect' route involves a direct connection of the motor cortex with the STN that does not involve the striatum. This circuit transmits quick cortical messages to the STN, and produces rapid, diffuse inhibition via the STN-GPi circuit - which is suggested to be a 'global stop' signal to inhibit the initiation of movement too soon or inappropriately [12].

Table 1. *Summary of basal ganglia pathways, their neurotransmitter profiles, and net effects on thalamo-cortical motor output.*

Pathway	Key Nodes	Neurotransmitters	Dopamine Receptor	Net Motor Effect
Direct	Striatum (D1-MSN) → GPi/SNr → Thalamus	GABA (inhibitory at GPi)	D1 (excitatory)	Movement facilitation
Indirect	Striatum (D2-MSN) → GPe → STN → GPi	GABA; Glutamate (STN)	D2 (inhibitory)	Movement suppression

Hyperdirect	Motor Cortex → STN → GPi/SNr	Glutamate	N/A	Rapid global inhibition
Nigrostriatal	SNc → Striatum	Dopamine	D1, D2	Modulates both pathways

Cerebellar Circuitry

The cerebellum, which occupies about 10 percent of the brain volume, yet controls over half of all the CNS neurons, is the chief error-correcting motor system engine [13]. Its major role is to compare the desired motor signals to real sensorimotor signals and issue corrective signals- a process that is referred to as, internal model computation.

The cerebellar cortex is stratified into three layers, which are the molecular layer (comprising parallel fibers and Purkinje cells dendrites), the Purkinje cell layer (comprising of large GABAergic Purkinje cells single layer), and the granule cell layer (comprising of densely packed granule cells and Golgi interneurons). Purkinje cells form the only output of cerebellar cortex, which send out inhibitory impulses to the deep cerebellar nuclei (dentate, interposed, and fastigial nuclei) [14].

The cerebellum also has two major inputs: mossy fibers (spinal cord, brainstem, and cerebrocortical connections) which synapses with the granule cells, and climbing fibers (only contralateral on inferior olive) which synapses with the Purkinje cell dendrites. Climbing fibers are thought to transmit some form of teaching signal (which encodes movement errors) that stimulates long-term depression (LTD) at parallel fiber-Purkinje cell synapses - the cellular mechanism of cerebellar motor learning [15].

The cerebellum aromatizes towards the motor cortex through dentate nucleus-thalamus-cortex relay. This error-correction loop in movement disorders of the cerebellum is dysfunctional, resulting in ataxia (imprecise movements of the limbs), dysdiadochokinesia, gait ataxia, and intention tremor.

Brainstem and Spinal Circuits

In addition to the major subcortical nuclei and the cortex, there are other brainstem structures that are very important in controlling motor behaviors. The pedunculopontine nucleus (PPN) - at the cross-over in the midbrain and pons - is involved in locomotion and postural regulation and arousal. Its mutual connectivity with the basal ganglia and its impairment in Parkinson disease due to which gait freezing and postural instability occur [16]. The reticular formation is full of propriospinal neurons that coordinate axial and proximal limb movements and the vestibular nuclei process the necessary axial and proximal limb movements that are necessary to control the balance.

The alpha motor neurons at the spinal cord level are connected to the corticospinal, rubrospinal, reticulospinal and vestibulospinal tracts of the CNS, and the segmental interneurons (Ia inhibitory interneurons and Renshaw cells). Spinal interneuronal networks mediate reciprocal, recurrent, and reflexive arcs that are associated with the pathophysiology of spasticity, rigidity and hyperreflexia in a variety of movement disorders [17].

Parkinson's Disease: Neurophysiology of the Hypokinetic State.

Dopaminergic Degeneration and Circuit Imbalance

Parkinson disease (PD) refers to the paradigmatic movement disorder called hypokinetic as it is characterized by the cardinal motor symptoms of bradykinesia (slowness of movement), rigidity (increased muscle tone), resting tremor, and postural instability. These characteristics are a result of successful degeneration of the dopaminergic neurons in the SNc causing serious depletion of dopamine in striatum [18].

Direct/indirect pathway model Via the direct/indirect pathway model, dopamine loss predicts and causes: (1) under-activation of the direct pathway by loss of D1 receptor stimulation, leading to impaired thalamic disinhibition; (2) excessively activated indirect pathway by loss of D2-mediated suppressions and therefore excessively excited STN-GPi, thereby increasing thalamic inhibition. The overall result is that there is less thalamocortical drive to motor cortex, which leads to the clinical state of hypokineticism [19, 20].

Nevertheless, this classical paradigm, although heuristically useful, is simplistic with regard to the real pathophysiology. Recent data indicate that PD entails complex dynamics of firing rates, firing patterns and synchronization throughout the basal ganglia network which cannot be described using rate models [21].

Oscillatory Dysfunction: The Beta Hypothesis.

The description of pathological beta-frequency (13-30 Hz) oscillations of the basal ganglia-cortical network has become one of the most important contributions to the knowledge of PD neurophysiology. The off-medication, hypokinetic state, intraoperative local field potential (LFP) recording of the STN in PD patients is regularly accompanied by a strong beta-band oscillatory activity which is dramatically suppressed by dopaminergic medication as well as by voluntary movement [22].

Beta oscillation strength is related to the intensity of bradykinesia and rigidity: the more severe in motor performance, the higher is the power of beta oscillations; and the inhibition of beta oscillation by dopaminergic stimulants or DBS was associated with improvement. This observation has given rise to the PD beta hypothesis that pathological BG-cortical loop BG beta synchrony subjects motor circuits to a functional brake on movement initiation, which is blocked by a network transition [23].

It is believed that the genesis of beta oscillations in PD entails a number of interacting mechanisms: (1) enhanced cortical beta rhythms provoked by hyperdopaminergic cortical states counterintuitive to hypodopaminergic striatal ones; (2) heightened synchronization of STN-GPe networks, possibly through reciprocal excitatory-inhibitory interactions (3) diminished gamma-band activity (>30 Hz) that under normal conditions facilitates movement [24]. The gamma oscillations, on the other hand, are related to dopaminergic activation, movement, as well as normal motor functioning.

Cellular Mechanisms: Altered Firing Patterns and Plasticity.

On a single neuron level, PD pathophysiology lies in drastic alterations in the firing pattern, independent of rate, between BGs. STN neurons of dopamine-depleted mice and PD patients have elevated burst firing (combined discharges separated by pauses), coordinated firing

with other neurons, and oscillatory firing (entrained to cortical beta rhythms). It is assumed that these changes interfere with the accurate timing coding needed to make a choice of action [25].

State-dependent activity is also altered in striatal medium spiny neurons (MSNs). In the normal conditions, MSNs move between, in hyperpolarized condition, known as a down state, and in depolarized condition known as up state near firing threshold, which is mediated by cortical inputs. The loss of dopamine disturbs this bistability, especially in D2-expressing indirect pathway MSNs, making them hyperexcitable, augmenting the indirect pathway hyperactivity [26].

PD has a tremendous influence on synaptic plasticity. At corticostriatal synapses - the cellular basis of procedural motor learning, dopamine is needed in long-term potentiation (LTP) and long-term depression (LTD). Without it, plasticity will result in aberrant or complete failure, which also adds to motor learning impairments and motor adaptation impairments that are typical of PD [27].

Alpha-Synuclein, Neuroinflammation, and Network Spread.

Lewy bodies and Lewy neurites are the pathological hallmarks of PD with the accumulation of misfolded alpha-synuclein. The spreading of alpha-synuclein pathology can be predicted by Braake staging: it starts in the olfactory bulb and dorsal motor nucleus of the vagus (stage 1), extends to locus coeruleus (stage 2), SNc (stage 3-4), limbic areas (stage 5-6) [28]. This staging explains why the non-motor prodromal elements of PD (hyposmia, constipation, REM sleep behavior disorder) are years or decades before the onset of motor symptoms.

Dopaminergic neurodegeneration is increased by neuroinflammation guided by activated microglia that releases pro-inflammatory cytokine (TNF-alpha, IL-1beta, IL-6). Further acceleration of neuronal loss is caused by mitochondrial dysfunction and oxidative stress, especially in the dopaminergic neurons of SNc that are metabolically vulnerable because the high pacemaking activity is accompanied by a low mitochondrial buffering capacity [29].

Essential Tremor: Cerebellar Network Oscillations

Clinical and Neurophysiological Overview

The patient presents with a history of a stroke, which has since been confirmed by computerized tomography (CT) and magnetic resonance imaging (MRI) (Armstrong et al., 2016).
The patient has a history of a stroke that has been further supported by computerized tomography (CT) and magnetic resonance imaging (MRI) (Armstrong et al., 2016).

The most typical adult movement disorder is essential tremor (ET) which is predominantly a postural and kinetic tremor of mainly upper extremities but may extend to include the head, voice, and lower extremities. Although quite common, the neurophysiology of ET was not well understood until quite recently [30].

Neurophysiologically, tremor in ET is a kinetic tremor (peak of sustained posture and movement) between frequencies of 4-12 Hz, which is different than the resting tremor of 3-6 Hz in PD. Surface electromyography (EMG) demonstrates antagonist muscle pair oscillatory activity that is synchronized in time; a characteristic of a central oscillator operating to express peripheral tremor [31].

The Cerebellum as Central Oscillator.

The latest data strongly points to the olivocerebellar system as the main cause of tremor vibrations in ET. Studies of positron emission tomography (PET), and functional MRI (fMRI) all show an enhancement of cerebellar activity during tremor in ET patients. Besides, lesions of the ventral intermediate nucleus of the thalamus (VIM), the main thalamic relay of cerebellar output, are reliably associated with elimination of tremor, which makes the cerebello-thalamo-cortical circuit the primary tremor-generating loop [32].

The inferior olive (IO) that forms climbing fiber inputs to Purkinje cells possesses internal pacemaking capability through low-threshold calcium currents and electrical coupling of IO neurons through gap junctions. This linkage of electrical activity enables the synchronous oscillation of IO neurons at tremor frequencies that may initiate coherent oscillatory activity throughout the cerebellar cortex and extend into the cerebello-thalamo-cortical circuit. Harmaline - a drug that significantly

enhances IO oscillatory activity - tremor forms in animal models after the administration of harmaline, which is similar to ET giving further evidence in support of this model [33].

ET mouse models/ET patients recordings of the purkinje cell activity suggest that abnormal simple spike firing and higher complex spike firing to the presence of pathological olivary input interferes with the normal production of the purkinje cell and destabilizes the output signal of the cerebellar signal. More importantly, GABAergic interneurons within the cerebellar cortex - basket cells, stellate cells, and Golgi cells, exhibit a lower post-mortem tissue density that may disrupt the feedforward inhibition and allow pathological oscillatory activity to spread [34].

Differential Neurophysiology: ET vs. PD Tremor.

Clinical differences between ET and the resting tremor of PD are of importance, but may be neurophysiologically subtle. Among the differences: ET tremor is postural/kinetic, cerebellar, whereas PD tremor is primarily at rest, and is associated with BG-thalamocortical re-entrant loops. Dopaminergic agents have a strong antitremor effect in PD, but are not very effective in ET. PD tremor oscillatory activity at 4-6 Hz is associated with thalamic tremor cells that become synchronized with peripheral tremor, as opposed to the cerebellar output higher frequencies which drive ET [35].

Table 2. *Neurophysiological comparison of major tremor syndromes.*

Feature	Essential Tremor	Parkinson's Disease Tremor	Cerebellar Tremor
Frequency (Hz)	4–12 Hz	3–6 Hz	2–4 Hz
Tremor type	Postural / Kinetic	Resting (pill-rolling)	Intention / Kinetic
Primary generator	Cerebello-olivary loop	BG-thalamocortical loop	Cerebellar dentate nucleus

Thalamic relay	VIM nucleus	VIM / VA-VL nuclei	VIM nucleus
Response to levodopa	Minimal	Significant	None
Characteristic lesion	Reduced Purkinje cells, IO hyperactivity	SNc dopaminergic loss	Dentate nucleus / cerebellar cortex
DBS target	VIM thalamus	STN or GPi	VIM thalamus

Huntington's Disease: Striatal Neurodegeneration and Network Dysfunction

Genetic Basis and Pathophysiology

Huntington's disease (HD) is an autosomal dominant neurodegenerative disorder caused by an expanded CAG trinucleotide repeat in the HTT gene on chromosome 4p16.3, encoding an abnormally long polyglutamine tract in the huntingtin protein. Repeat lengths above 40 cause full penetrance, with longer expansions correlating with earlier age of onset [36]. Mutant huntingtin (mHtt) acquires toxic gain-of-function properties while also losing normal huntingtin's neuroprotective functions, triggering a cascade of transcriptional dysregulation, mitochondrial dysfunction, excitotoxicity, and ultimately selective neuronal death.

The striatum is the primary site of neurodegeneration in HD, with preferential early loss of indirect pathway MSNs (D2-expressing, enkephalin-positive) in the caudate nucleus and putamen. As disease progresses, direct pathway MSNs and neurons in the cerebral cortex, cerebellum, and other subcortical structures are also lost [37].

Choreiform Hyperkinesia: Neurophysiological Mechanism

The early motor manifestation of HD is chorea — involuntary, random, flowing movements that flit from one body part to another, resembling fragments of voluntary movement. The neurophysiological basis of chorea is explained directly by the direct/indirect pathway model: preferential loss of indirect pathway MSNs removes inhibitory drive from GPe, disinhibiting the STN-GPi axis is abolished. The relative

preservation of direct pathway MSNs means the GPi receives increased inhibitory drive, further reducing GPi output. The result is pathological disinhibition of the thalamus, producing excessive, unrestrained thalamocortical activation — the neurophysiological substrate of chorea [38].

As HD progresses and direct pathway MSNs are also lost, the balance shifts: both pathways are now depleted, and the emerging clinical picture transitions from hyperkinesia to hypokinesia, rigidity, and cognitive decline — particularly prominent in juvenile-onset HD with very long CAG repeats [39].

Cortical Excitability and Synaptic Pathology

In addition to striatal degeneration, HD has major pathophysiology in cortex. Examining TMS in the HD patients show that cortical excitability is augmented whereas cortical inhibition is compromised (diminished short interval cortical inhibition, SICI) in agreement with the hyperkinetic condition [40]. The source of corticospinal projections are the layer V pyramidal neurons of motor cortex which is affected by dendritic atrophy and loss of synapses, which causes motor cortex dysfunction.

A suggested central mechanism of MSN death in HD is excitotoxicity, which is a result of excessive glutamate activation of striatal NMDA receptors. The striatum is bombarded by corticostriatal glutamatergic projections and the mutant huntingtin proceeds to sensitize NMDA receptors to calcium-mediated excitotoxic cell death. The early pathophysiological evidence of abnormal calcium signaling in MSNs is observed in disease models before neurodegeneration occurs [41].

Dystonia: Maladaptive Plasticity and Loss of Motor Inhibition

Defining Features and Circuit Hypotheses.

Dystonia is a disorder of movement that involves persistent or intermittent spasmodic contractions of muscles that result in abnormal muscle and posture movements, repetitive movements. Either it can be focal (e.g., cervical dystonia, writer cramp, blepharospasm) or segmental or generalized. Although they may be diverse in terms of etiology,

dystonias have some common neurophysiological characteristics that imply a common pathophysiological process [42].

The main neurophysiological changes of dystonia are: (1) loss of surround inhibition - loss of the capacity to inhibit the activity of muscles around a desired movement appropriately; (2) loss of reciprocal inhibition at the spinal cord level; (3) elevated cortical excitability and reduced GABAergic inhibiatory tone; (4) aberrant motor learning and maladaptive plasticity at corticostriatal synapses [43].

Basal Ganglia, Cerebellum, and Spinal Cord Involvement.

The use of DBS recording documents and functional neuroimaging has determined that dystonia entails impairment at various levels of the motor neuraxis. GPi neurons of dystonic patients exhibit lower firing rates and erratic, phasic and grouped discharges with abnormal low-frequency oscillations (4-12 Hz) instead of the normal irregular high-frequency firing in the basal ganglia [44]. This disturbed output is a consequence of interfering with the thalamocortical signaling, which adds to the movement execution loss of spatial specificity.

Dystonia has been identified to have cerebral involvement in the brain. Motor calibration and updating of internal models normally happens in the cerebellum. Abnormal cerebellar activity in dystonia could cause unwanted revolving conduct via cerebello-thalamo-cortical loops. A cerebellar role is supported by lesion studies, animal models of dystonia and the effectiveness of cerebellar targeting therapies [45].

Among the neurons in the spinal cord, deactivation of gamma motor neurons enhances the intrafusal sensitivity of the spindle of the muscle, increases sensory feedback, and adds to the persistent co-contraction of the agonist and antagonist muscle characteristic of dystonic posturing [46].

Maladaptive Plasticity: The 'Excessive LTP' Model.

Among the most interesting neurophysiological explanations of dystonia is the assumption that the disorder is caused by too much or too soon-induced potentiation of corticostriatal synapses through long-term potentiation (LTP). According to this model, sensorimotor training (in the case of a musician, athlete, or typist) causes pathological instead of adaptive plasticity in genetically susceptible individuals and imprints them with motor patterns that are abnormal. TMS studies demonstrate that cortical plasticity (measured by paired associative stimulation

protocols) is exaggerated in focal dystonia patients and abnormally persists, consistent with impaired homeostatic plasticity mechanisms [47].

This model explains both the task-specificity of many focal dystonias (writer's cramp, musician's cramp) and the therapeutic potential of sensorimotor re-training and botulinum toxin, which by temporarily reducing afferent input from affected muscles may allow maladaptive plasticity to partially reverse.

Cerebellar Ataxias: Error Correction Failure

Classification and General Neurophysiology

The cerebellar ataxias are a heterogeneous group of disorders — including spinocerebellar ataxias (SCAs), Friedreich's ataxia, multiple system atrophy cerebellar type (MSA-C), and sporadic/idiopathic late-onset ataxia — unified by cerebellar dysfunction producing the clinical ataxia syndrome [48]. Common features include gait ataxia, limb dysmetria, dysdiadochokinesia, scanning dysarthria, and nystagmus.

Neurophysiologically, cerebellar ataxia reflects disruption of the cerebellum's role as an internal model and error-corrector. The inability to predict sensory consequences of motor commands and generate corrective forward models leads to the characteristic movement oscillations, dysmetria, and decomposition of movement [49].

Purkinje Cell Dysfunction and Cerebellar Output Failure

In most cerebellar ataxias, Purkinje cells are selectively vulnerable. Loss of Purkinje cells removes inhibitory drive from the deep cerebellar nuclei, particularly the dentate nucleus, disinhibiting cerebellar output. But the main loss is not that of surplus production, but that of a finer, task-specific modulation of deep cerebellar nuclear activity that is available through Purkinje cells to the rest of the brain [50].

In SCA1, SCA2, SCA3 and other polyglutamine ataxias, the mutant proteins build up in Purkinje cells and other cerebellar neurons, and they interfere with protein quality regulation, gene expression and eventually lead to cell death. In every type of SCA, different molecular pathways meet at the disruption of the physiology of Purkinje cells; namely, altered calcium channels (P/Q-type in SCA6), impaired

metabotropic glutamate receptor signalling (mGluR1 in SCA1) and disturbed axonal transport [51].

Olivocerebellar Timing and Rhythm Generation.

One of these, the inferior olive (IO), is involved in both the error signals of climbing fibers and the rhythm generation of the cerebellum, though electrical coupling. In those diseases that involve the olivocerebellar pathway, the impairment of cerebellar adaptation and motor learning due to the loss of precise error-correction signaling continues ataxic movements. The rhythmic characteristics of the IO may also be pathologic - that is the production of low-frequency oscillations, which will give the tremorous component (palatal tremor, Holmes tremor) of some ataxic syndromes related to hypertrophic olivary degeneration [52].

Neurophysiological Biomarkers and Neuromodulation

Electrophysiological Biomarkers

Electrophysiological Biomarkers.--Electrophysiological biomarkers include the pH gradient across the lipid membrane (PLM) and the membrane potential (Vm) ions.<|human|>Electrophysiological Biomarkers.--Electrophysiological biomarkers are the pH gradient across lipid membrane (PLM) and the membrane potential (Vm) ions.

Comprehensive description of pathological oscillatory signatures of movement disorders has stimulated the desire to develop neurophysiological biomarkers to diagnose, stage disease and follow-up treatment. STN-LFPs (measured through externalized leads of the DBS) have a strong correlation between beta-band power and the severity of motor impairment in PD, and responds predictably to medication state [53].

The movement-related cortical potential (MRCP) indexes of EEG-derived measures (such as movement-related cortical potentials (MRCPs), corticomuscular coherence, event-related desynchronization (ERD), and cross-frequency coupling between beta and gamma bands are being considered as non-invasive biomarkers in PD, ET, and dystonia. The example of pathological beta-gamma coupling that is present in PD and in which the amplitude of the gamma activity is

modulated by the phase of beta oscillations can be considered a remarkably strong indicator of disease severity [54].

Deep Brain Stimulation Mechanisms.

The most effective and the most common neuromodulatory treatment in advanced PD (STN or GPi target), essential tremor (VIM target), including dystonia (GPi target), and other movement disorders, is the deep brain stimulation (DBS) high-frequency electric stimulation (usually 130-185 Hz) via surgically placed electrodes [55]. The mechanisms by which DBS exerts therapeutic benefit have been intensively investigated.

Early hypotheses posited that high-frequency stimulation inhibits local neurons at the stimulation site (functional lesion hypothesis). However, subsequent evidence demonstrated that DBS also activates axons — including both efferent projections from the stimulated nucleus and passing fiber tracts — generating orthodromic and antidromic impulses that propagate throughout connected networks [56]. Current understanding favors a 'network modulation' model: DBS disrupts pathological oscillatory synchrony within BG-cortical circuits, overriding pathological beta oscillations with high-frequency, desynchronized activity that more closely approximates the physiological 'asynchronous' state associated with movement.

Support for this hypothesis comes from studies demonstrating that DBS-induced motor improvement correlates with beta desynchronization and that stimulation parameters producing optimal clinical benefit are also those most effective at disrupting beta oscillatory activity [57].

Closed-Loop Neurostimulation

Conventional DBS delivers continuous stimulation regardless of the patient's motor state, resulting in both suboptimal therapeutic efficacy during periods of lower pathological activity and unnecessary stimulation-related side effects. The identification of neurophysiological biomarkers — particularly STN beta-band power — has enabled the development of 'adaptive' or 'closed-loop' DBS systems that automatically adjust stimulation parameters in response to real-time neural signals [58].

Clinical trials of closed-loop DBS in PD patients have shown that beta-triggered stimulation has the same or better motor advantages than

conventional continuous DBS and uses 40-50-less stimulation energy, which has potential to increase battery life and decrease side effects [59]. This sense-and-stimulate type of model is a paradigm shift in neuromodulation, and analogs are being developed to ET (tremor-triggered stimulation) and dystonia (movement-triggered stimulation).

Multimodal biomarkers - integrating LFP oscillations with neural firing patterns, kinematic measurements and wearable sensor data - with machine learning algorithms could be added in the future to optimize therapy in patients with a wide range of motor and non-motor disease expressions, which real-time responds to changes in disease state throughout the day [60].

Emerging Concepts and Future Directions.

There are a number of emerging concepts which are transforming our knowledge of neurophysiology of movement disorders. First, the non-motor dimensions of BG-cortical circuit dysfunction are increasingly appreciated: cognitive, psychiatric, and autonomic manifestations of PD, HD, and dystonia reflect pathological involvement of limbic and associative corticobasal ganglia loops, not merely the sensorimotor circuit [61]. This recognition is driving the development of adaptive DBS systems that can target multiple circuit dimensions simultaneously.

Second, the contribution of peripheral mechanisms — including sensory afferent processing, proprioceptive feedback dysregulation, and neuromuscular junction changes — to the clinical expression of central movement disorders is being reconsidered. Particularly in dystonia, abnormal sensorimotor integration at cortical and subcortical levels appears to be as fundamental as purely motor circuit dysfunction [62].

Third, advances in single-unit and population-level recording technologies — including high-density silicon probes, calcium imaging in behaving animals, and human intraoperative single-unit recordings — are providing unprecedented resolution of circuit dynamics in movement disorders. These technologies, combined with optogenetic dissection of specific cell-type and pathway contributions, are enabling the construction of biologically realistic computational models that can predict the effects of targeted interventions [63].

Lastly, biomarker-based early intervention measures - that is, using prodromal neurophysiological indicators of individuals at risk of

PD and HD prior to motor symptom onset - is also a significant direction. It would be possible to identify pathological beta oscillations or cortical plasticity defects many years prior to diagnosis and then intervene neuroprotectively when the neuronal substrate may be better salvaged [64].

Conclusion

Neurophysiology of movement disorders has evolved into a field with an interest in discovering the sites of lesions to a complex scientific domain of studying dynamic network dysfunctions on a molecular, cellular, and circuit scale. Extended by an in-depth grasp of oscillatory dynamics, synaptic plasticity processes, cerebellar error-corrective pathways, and spinal inhibitory networks, the classical direct/indirect pathway account of how the basal ganglia are related to diverse phenomenology of movement disorders describes a potent framework of understanding the diverse phenomenology of movement disorders.

The case of Parkinson disease demonstrates how the process of dopaminergic depletion leads to pathological beta synchrony, which functionally imposes a functional block on movement initiation, how the oscillations of the olivocerebellar network contribute to peripheral tremor, how senior degeneration in the striatum can lead to the establishment of pathological movement patterns, and how the loss of surround inhibition leads to the maladaptive cortical plasticity seen in dystonia, and the catastrophic impact of losing the main error-correction system found in the cerebellum.

Importantly, the study of these processes is not just an intellectual pursuit: it directly contributes to the formulation of the pharmacological, surgical and neuromodulative treatments and optimization. Adaptive, closed-loop, real-time neurophysiological biomarker-based DBS systems are perhaps the most direct application of fundamental neurophysiology to clinical care and a sign of the future of precision medicine, especially in movement disorders, on a case-by-case basis.

Continued studies - especially the incorporation of genetics, molecular biology and further electrophysiology and computational modeling - are likely to further qualify this knowledge, new therapeutic

targets, and eventually, lead to disease-modifying interventions that go beyond symptomatic treatment to the neurophysiological basis.

References

[1] Fahn S. Classification of movement disorders. Mov Disord. 2011;26(6):947-57.

[2] Louis ED, Ferreira JJ. How common is the most common adult movement disorder? Update on the worldwide prevalence of essential tremor. Mov Disord. 2010;25(5):534-41.

[3] GBD 2016 Parkinson's Disease Collaborators. Global, regional, and national burden of Parkinson's disease, 1990-2016: a systematic analysis for the Global Burden of Disease Study 2016. Lancet Neurol. 2018;17(11):939-53.

[4] Hammond C, Bergman H, Brown P. Pathological synchronization in Parkinson's disease: networks, models and treatments. Trends Neurosci. 2007;30(7):357-64.

[5] Silberstein P, Kuhn AA, Kupsch A, et al. Patterning of globus pallidus local field potentials differs between Parkinson's disease and dystonia. Brain. 2003;126(Pt 12):2597-608.

[6] Penfield W, Boldrey E. Somatic motor and sensory representation in the cerebral cortex of man as studied by electrical stimulation. Brain. 1937;60(4):389-443.

[7] Cunnington R, Iansek R, Bradshaw JL, Phillips JG. Movement-related potentials in Parkinson's disease. Brain. 1995;118(Pt 4):935-50.

[8] Middleton FA, Strick PL. Basal ganglia and cerebellar loops: motor and cognitive circuits. Brain Res Rev. 2000;31(2-3):236-50.

[9] DeLong MR. Primate models of movement disorders of basal ganglia origin. Trends Neurosci. 1990;13(7):281-5.

[10] Albin RL, Young AB, Penney JB. The functional anatomy of basal ganglia disorders. Trends Neurosci. 1989;12(10):366-75.

[11] Gerfen CR, Surmeier DJ. Modulation of striatal projection systems by dopamine. Annu Rev Neurosci. 2011;34:441-66.

[12] Nambu A, Tokuno H, Takada M. Functional significance of the cortico-subthalamo-pallidal 'hyperdirect' pathway. Neurosci Res. 2002;43(2):111-7.

[13] Ito M. Cerebellar circuitry as a neuronal machine. Prog Neurobiol. 2006;78(3-5):272-303.

[14] Eccles JC, Ito M, Szentagothai J. The Cerebellum as a Neuronal Machine. Berlin: Springer; 1967.

[15] Marr D. A theory of cerebellar cortex. J Physiol. 1969;202(2):437-70.

[16] Pahapill PA, Lozano AM. The pedunculopontine nucleus and Parkinson's disease. Brain. 2000;123(Pt 9):1767-83.

[17] Nielsen JB, Crone C, Hultborn H. The spinal pathophysiology of spasticity — from a basic science point of view. Acta Physiol. 2007;189(2):171-80.

[18] Bernheimer H, Birkmayer W, Hornykiewicz O, et al. Brain dopamine and the syndromes of Parkinson and Huntington. J Neurol Sci. 1973;20(4):415-55.

[19] Mink JW. The basal ganglia: focused selection and inhibition of competing motor programs. Prog Neurobiol. 1996;50(4):381-425.

[20] Obeso JA, Rodriguez-Oroz MC, Benitez-Temino B, et al. Functional organization of the basal ganglia: therapeutic implications for Parkinson's disease. Mov Disord. 2008;23(Suppl 3):S548-59.

[21] Surmeier DJ, Ding J, Day M, Wang Z, Shen W. D1 and D2 dopamine-receptor modulation of striatal glutamatergic signaling in striatal medium spiny neurons. Trends Neurosci. 2007;30(5):228-35.

[22] Brown P, Oliviero A, Mazzone P, et al. Dopamine dependency of oscillations between subthalamic nucleus and pallidum in Parkinson's disease. J Neurosci. 2001;21(3):1033-8.

[23] Kühn AA, Kupsch A, Schneider GH, Brown P. Reduction in subthalamic 8-35 Hz oscillatory activity correlates with clinical improvement in Parkinson's disease. Eur J Neurosci. 2006;23(7):1956-60.

[24] Tinkhauser G, Pogosyan A, Little S, et al. The modulatory effect of adaptive deep brain stimulation on beta bursts in Parkinson's disease. Brain. 2017;140(4):1053-67.

[25] Bergman H, Wichmann T, DeLong MR. Reversal of experimental parkinsonism by lesions of the subthalamic nucleus. Science. 1990;249(4975):1436-8.

[26] Kreitzer AC, Malenka RC. Striatal plasticity and basal ganglia circuit function. Neuron. 2008;60(4):543-54.

[27] Picconi B, Centonze D, Hakansson K, et al. Loss of bidirectional striatal synaptic plasticity in L-DOPA-induced dyskinesia. Nat Neurosci. 2003;6(5):501-6.

[28] Braak H, Del Tredici K, Rüb U, et al. Staging of brain pathology related to sporadic Parkinson's disease. Neurobiol Aging. 2003;24(2):197-211.

[29] Tansey MG, Wallings RL, Houser MC, Herrick MK, Keating CE, Joers V. Inflammation and immune dysfunction in Parkinson disease. Nat Rev Immunol. 2022;22(11):657-73.

[30] Louis ED. Essential tremor. N Engl J Med. 2001;345(12):887-91.

[31] Elble RJ, Koller WC. Tremor. Baltimore: Johns Hopkins University Press; 1990.

[32] Benabid AL, Pollak P, Gervason C, et al. Long-term suppression of tremor by chronic stimulation of the ventral intermediate thalamic nucleus. Lancet. 1991;337(8738):403-6.

[33] Llinas R, Volkind RA. The olivo-cerebellar system: functional properties as revealed by harmaline-induced tremor. Exp Brain Res. 1973;18(1):69-87.

[34] Louis ED, Faust PL, Vonsattel JP, et al. Neuropathological changes in essential tremor: 33 cases compared with 21 controls. Brain. 2007;130(Pt 12):3297-307.

[35] Deuschl G, Raethjen J, Lindemann M, Krack P. The pathophysiology of tremor. Muscle Nerve. 2001;24(6):716-35.

[36] The Huntington's Disease Collaborative Research Group. A novel gene containing a trinucleotide repeat that is expanded and unstable on Huntington's disease chromosomes. Cell. 1993;72(6):971-83.

[37] Vonsattel JP, DiFiglia M. Huntington disease. J Neuropathol Exp Neurol. 1998;57(5):369-84.

[38] Penney JB Jr, Vonsattel JP, MacDonald ME, Gusella JF, Myers RH. CAG repeat number governs the development rate of pathology in Huntington's disease. Ann Neurol. 1997;41(5):689-92.

[39] Squitieri F, Frati L, Ciarmiello A, Lastoria S, Quarrell O. Juvenile Huntington's disease: does a dosage-effect pathogenic mechanism differ from the classical adult disease? Mech Ageing Dev. 2006;127(2):208-12.

[40] Lefaucheur JP. Motor cortex dysfunction revealed by cortical excitability studies in Parkinson's disease: influence of antiparkinsonian treatment and cortical stimulation. Clin Neurophysiol. 2005;116(2):244-53.

[41] Raymond LA, André VM, Cepeda C, Gladding CM, Milnerwood AJ, Levine MS. Pathophysiology of Huntington's disease: time-dependent alterations in synaptic and receptor function. Neuroscience. 2011;198:252-73.

[42] Albanese A, Bhatia K, Bressman SB, et al. Phenomenology and classification of dystonia: a consensus update. Mov Disord. 2013;28(7):863-73.

[43] Hallett M. Pathophysiology of dystonia. J Neural Transm Suppl. 2006;70:485-8.

[44] Starr PA, Rau GM, Davis V, et al. Spontaneous pallidal neuronal activity in human dystonia: comparison with Parkinson's disease and normal macaque. J Neurophysiol. 2005;93(6):3165-76.

[45] Neychev VK, Fan X, Mitev VI, Hess EJ, Bhatt DL. The basal ganglia and cerebellum interact in the expression of dystonic movement. Brain. 2008;131(Pt 9):2499-509.

[46] Quartarone A, Pisani A. Abnormal plasticity in dystonia: Disruption of synaptic homeostasis. Neurobiol Dis. 2011;42(2):162-70.

[47] Quartarone A, Bagnato S, Rizzo V, et al. Abnormal associative plasticity of the human motor cortex in writer's cramp. Brain. 2003;126(Pt 12):2586-96.

[48] Ashizawa T, Xia G. Ataxia. Continuum (Minneap Minn). 2016;22(4):1208-26.

[49] Wolpert DM, Miall RC, Kawato M. Internal models in the cerebellum. Trends Cogn Sci. 1998;2(9):338-47.

[50] Heckroth JA. Quantitative morphological analysis of the cerebellar nuclei in normal and lurcher mutant mice. J Comp Neurol. 1994;343(1):173-82.

[51] Paulson HL, Shakkottai VG, Clark HB, Bhatt DL. Polyglutamine spinocerebellar ataxias — from genes to potential treatments. Nat Rev Neurosci. 2017;18(10):613-26.

[52] Goyal M, Versnick E, Tuite P, et al. Hypertrophic olivary degeneration: metaanalysis of the temporal evolution of MR findings. AJNR Am J Neuroradiol. 2006;27(8):1675-82.

[53] Priori A, Foffani G, Rossi L, Marceglia S. Adaptive deep brain stimulation (aDBS) controlled by local field potential oscillations. Exp Neurol. 2013;245:77-86.

[54] de Hemptinne C, Swann NC, Ostrem JL, et al. Therapeutic deep brain stimulation reduces cortical phase-amplitude coupling in Parkinson's disease. Nat Neurosci. 2015;18(5):779-86.

[55] Benabid AL, Chabardes S, Mitrofanis J, Pollak P. Deep brain stimulation of the subthalamic nucleus for the treatment of Parkinson's disease. Lancet Neurol. 2009;8(1):67-81.

[56] McIntyre CC, Grill WM. Extracellular stimulation of central neurons: influence of stimulus waveform and frequency on neuronal output. J Neurophysiol. 2002;88(4):1592-604.

[57] Kühn AA, Tsui A, Aziz T, et al. Pathological synchronisation in the subthalamic nucleus of patients with Parkinson's disease relates to both bradykinesia and rigidity. Exp Neurol. 2009;215(2):380-7.

[58] Little S, Pogosyan A, Neal S, et al. Adaptive deep brain stimulation in advanced Parkinson disease. Ann Neurol. 2013;74(3):449-57.

[59] Arlotti M, Marceglia S, Foffani G, et al. Eight-hours adaptive deep brain stimulation in patients with Parkinson disease. Neurology. 2018;90(11):e971-6.

[60] Herron JA, Thompson MC, Brown T, et al. Chronic electrocorticography for sensing movement intention and closed-loop deep brain stimulation with wearable sensors in an essential tremor patient. J Neurosurg. 2017;127(3):580-7.

[61] Haber SN, Calzavara R. The cortico-basal ganglia integrative network: the role of the thalamus. Brain Res Bull. 2009;78(2-3):69-74.

[62] Abbruzzese G, Berardelli A. Sensorimotor integration in movement disorders. Mov Disord. 2003;18(3):231-40.

[63] Nir Y, Massimini M, Boly M, Tononi G. Sleep and consciousness. In: Cavanna AE, Nani A, Blumenfeld H, Laureys S, eds. Neuroimaging of Consciousness. Berlin: Springer; 2013. p. 133-82.

[64] Postuma RB, Berg D. Advances in markers of prodromal Parkinson disease. Nat Rev Neurol. 2016;12(11):622-34.

******

ISBN: 978-93-7143-814-8

Chapter - 5

Multithreading in Java

Ms. Rashim Rana
Assistant Professor (Department of Data Science)
ABES Engineering College, Ghaziabad
Email I'd: rashimrana02@gmail.com

ABSTRACT

Multithreading is one of the most powerful and complex features of the Java programming language, enabling concurrent execution of two or more parts of a program for optimal utilization of CPU resources. This chapter provides a comprehensive exploration of multithreading in Java, beginning with foundational concepts such as processes, threads, and the Java Memory Model. It systematically covers thread lifecycle management, synchronization mechanisms, inter-thread communication, and the advanced concurrency utilities provided by the java.util.concurrent package introduced in Java 5. The chapter also examines common pitfalls such as deadlocks, race conditions, and thread starvation, alongside best practices for designing robust concurrent applications. Through annotated code examples and conceptual diagrams, readers will gain the theoretical grounding and practical skills needed to implement efficient, thread-safe Java programs suitable for modern multi-core hardware environments.

Keywords: Multithreading, Java concurrency, Thread lifecycle, Synchronization, Deadlock, Race condition, java.util.concurrent, ExecutorService, Thread safety, Java Memory Model, Volatile keyword, Locks, Fork/Join framework

1. Introduction

Modern computing environments are overwhelmingly concurrent. From web servers handling thousands of simultaneous requests to desktop applications maintaining responsive user interfaces while performing background computations, concurrency is no longer an optional feature

— it is a necessity. Java, designed from its inception as a platform-independent, general-purpose language, made concurrency a first-class concern by embedding thread support directly into the language specification and the Java Virtual Machine (JVM) [1].

Multithreading refers to the ability of a CPU or a single core in a multi-core processor to provide multiple threads of execution concurrently. In Java, a thread is the smallest unit of processing that can be performed in an operating system (OS). Java provides built-in support for multithreading through the java.lang.Thread class and the Runnable interface, and subsequently through the more sophisticated java.util.concurrent package [2].

This chapter covers the end-to-end landscape of multithreading in Java. Section 2 establishes background concepts; Section 3 presents thread creation and lifecycle; Section 4 details synchronization; Section 5 explains inter-thread communication; Section 6 introduces advanced concurrency utilities; Section 7 discusses common concurrency problems; and Section 8 summarizes best practices.

2. Background Concepts

2.1 Processes vs. Threads

A process is an independent program in execution with its own memory space, resources, and state. A thread, often called a lightweight process, exists within a process and shares the process's resources — including memory and open files — but maintains its own program counter, stack, and local variables [3].

The key distinction is resource sharing: threads within the same process share the heap and method area of the JVM, enabling fast communication but also introducing risks of data corruption when multiple threads access shared data without appropriate coordination.

2.2 Concurrency vs. Parallelism

Concurrency means that multiple computations are happening at the same time conceptually — they may be interleaved on a single CPU core. Parallelism means that computations genuinely run simultaneously on multiple hardware cores. Java multithreading supports both models: the

JVM scheduler manages concurrency on single-core systems, while the OS can schedule threads across multiple cores for true parallelism [4].

2.3 The Java Memory Model (JMM)

The Java Memory Model, defined in the Java Language Specification and revisited significantly in Java 5 via JSR-133, specifies the rules by which threads interact through main memory. Key principles of the JMM include [5]:

- Each thread has its own working memory (cache) that may hold copies of shared variables.
- Writes to a variable by one thread may not be immediately visible to another thread.
- Synchronization constructs establish happens-before relationships guaranteeing visibility and ordering.

3. Thread Creation and Lifecycle

3.1 Creating Threads

Java provides two primary approaches for creating threads [6]:

3.1.1 Extending the Thread Class

A class can extend java.lang.Thread and override the run() method:

```
public class MyThread extends Thread {
    @Override
    public void run() {
        System.out.println("Thread running: "
            + Thread.currentThread().getName());
    }
    public static void main(String[] args) {
        MyThread t = new MyThread();
        t.start();  // Calls run() in a new thread
    }
}
```

3.1.2 Implementing the Runnable Interface

Implementing Runnable is preferred because it separates task logic from thread management and allows the class to extend another class:

```
public class MyRunnable implements Runnable {
    @Override
    public void run() {
        System.out.println("Runnable running: "
            + Thread.currentThread().getName());
    }
    public static void main(String[] args) {
        Thread t = new Thread(new MyRunnable());
        t.start();
    }
}
```

3.1.3 Using Callable and Future

Unlike Runnable, Callable can return a result and throw checked exceptions. It is used with ExecutorService and Future [7]:

```
Callable<Integer> task = () -> {
    return 42;
};
ExecutorService executor = Executors.newSingleThreadExecutor();
Future<Integer> future = executor.submit(task);
System.out.println("Result: " + future.get());
executor.shutdown();
```

3.2 Thread Lifecycle

A Java thread passes through well-defined states during its lifetime. The Thread.State enumeration defines the following states [8]:

State	Description
NEW	Thread created but start() not yet called.
RUNNABLE	Thread is executing in the JVM or ready to execute, waiting for CPU scheduling.
BLOCKED	Thread is waiting to acquire a monitor lock held by another thread.
WAITING	Thread is waiting indefinitely for another thread to perform an action (e.g., Object.wait()).
TIMED_WAITING	Thread is waiting for a specified period (e.g., Thread.sleep(), Object.wait(timeout)).
TERMINATED	Thread has completed execution or was terminated abnormally.

3.3 Key Thread Methods

Java's Thread class and related constructs provide several important methods for controlling thread behaviour [9]:

- start(): Starts the thread, causing run() to be invoked in a new thread of execution.
- sleep(long millis): Pauses the current thread for the specified duration without releasing held locks.
- join(): Waits for the thread on which it is called to complete before the calling thread continues.
- interrupt(): Requests that a thread stop what it is doing; the thread must check Thread.interrupted() or handle InterruptedException.
- yield(): Hints to the scheduler that the current thread is willing to yield its turn to other threads of equal priority.

4. Synchronization

When multiple threads access shared mutable data concurrently, without proper coordination, the result is non-deterministic and potentially incorrect. Synchronization mechanisms in Java ensure that critical sections of code are accessed by only one thread at a time, preserving data integrity [10].

4.1 The synchronized Keyword

Java's intrinsic lock (monitor) mechanism is implemented through the synchronized keyword. It can be applied to methods or blocks:

```
public class Counter {
    private int count = 0;

    // Synchronized method
    public synchronized void increment() {
        count++;
    }

    // Synchronized block
    public void decrement() {
        synchronized (this) {
            count--;
        }
    }

    public int getCount() { return count; }
}
```

4.2 The volatile Keyword

The volatile keyword guarantees visibility: reads and writes to a volatile variable are always performed directly in main memory, not from a thread's local cache. It prevents instruction reordering for that variable but does not provide atomicity for compound operations [11]:

```
public class SharedFlag {
    private volatile boolean running = true;
```

```
    public void stop() {
        running = false;  // Immediately visible to all threads
    }

    public void run() {
        while (running) {
            // perform work
        }
    }
}
```

4.3 The java.util.concurrent.locks Package

The java.util.concurrent.locks package (introduced in Java 5) provides more flexible locking mechanisms than synchronized, including the ability to attempt non-blocking lock acquisition, interruptible locking, and timed locking [12]:

```
import java.util.concurrent.locks.ReentrantLock;

public class SafeCounter {
    private final ReentrantLock lock = new ReentrantLock();
    private int count = 0;

    public void increment() {
        lock.lock();
        try {
            count++;
        } finally {
            lock.unlock(); // Always release in finally block
```

```
        }
    }
}
```

4.4 Atomic Variables

The java.util.concurrent.atomic package provides classes such as AtomicInteger, AtomicLong, and AtomicReference that support lock-free, thread-safe operations on single variables using hardware-level compare-and-swap (CAS) instructions [13]:

```
import java.util.concurrent.atomic.AtomicInteger;

public class AtomicCounter {
    private final AtomicInteger count = new AtomicInteger(0);

    public void increment() {
        count.incrementAndGet();
    }

    public int getCount() {
        return count.get();
    }
}
```

5. Inter-Thread Communication

Threads often need to coordinate their activities. Java provides several mechanisms for inter-thread communication [14].

5.1 wait(), notify(), and notifyAll()

These Object class methods must be called within a synchronized context. wait() releases the lock and suspends the calling thread; notify() wakes one waiting thread; notifyAll() wakes all waiting threads:

```
public class ProducerConsumer {
    private final Object lock = new Object();
    private int data;
    private boolean hasData = false;

    public void produce(int value) throws InterruptedException {
        synchronized (lock) {
            while (hasData) lock.wait();
            data = value;
            hasData = true;
            lock.notifyAll();
        }
    }

    public int consume() throws InterruptedException {
        synchronized (lock) {
            while (!hasData) lock.wait();
            hasData = false;
            lock.notifyAll();
            return data;
        }
    }
}
```

5.2 BlockingQueue

The BlockingQueue interface in java.util.concurrent provides a thread-safe queue that blocks the producer when the queue is full and blocks the consumer when the queue is empty, greatly simplifying the producer-consumer pattern [15]:

```
BlockingQueue<Integer> queue = new LinkedBlockingQueue<>(10);


// Producer thread
new Thread(() -> {
    try { queue.put(42); } catch (InterruptedException e) { ... }
}).start();

// Consumer thread
new Thread(() -> {
    try { int val = queue.take(); } catch (InterruptedException e) { ... }
}).start();
```

6. Advanced Concurrency Utilities

6.1 ExecutorService and Thread Pools

Creating and destroying threads is expensive. Thread pools maintain a set of pre-created worker threads, reusing them for multiple tasks. The Executors factory class provides several common pool configurations [16]:

```
ExecutorService pool = Executors.newFixedThreadPool(4);


for (int i = 0; i < 10; i++) {
    final int taskId = i;
    pool.execute(() ->
        System.out.println("Task " + taskId + " on "
            + Thread.currentThread().getName()));
```

```
}
pool.shutdown();
```

6.2 The Fork/Join Framework

Introduced in Java 7, the Fork/Join framework is designed for parallelism by recursively breaking tasks into smaller subtasks (fork), computing them, then combining results (join). It uses a work-stealing algorithm to maximize CPU utilisation [17]:

```
class SumTask extends RecursiveTask<Long> {
    private long[] arr;
    private int lo, hi;
    static final int THRESHOLD = 1000;

    public Long compute() {
        if (hi - lo <= THRESHOLD) {
            long sum = 0;
            for (int i = lo; i < hi; i++) sum += arr[i];
            return sum;
        }
        int mid = (lo + hi) / 2;
        SumTask left  = new SumTask(arr, lo, mid);
        SumTask right = new SumTask(arr, mid, hi);
        left.fork();
        return right.compute() + left.join();
    }
}
```

6.3 CompletableFuture

CompletableFuture, introduced in Java 8, enables composable, non-blocking asynchronous programming. It supports chaining of operations and exception handling in a fluent API style [18]:

```
CompletableFuture.supplyAsync(() -> fetchData())
    .thenApply(data -> process(data))
    .thenAccept(result -> System.out.println(result))
    .exceptionally(ex -> {
        System.err.println("Error: " + ex.getMessage());
        return null;
    });
```

6.4 Synchronizers

The java.util.concurrent package provides several higher-level synchronizers for coordinating thread activities [19]:

- CountDownLatch: Allows one or more threads to wait until a set of operations being performed in other threads completes.
- CyclicBarrier: Lets a set of threads all wait for each other to reach a common barrier point.
- Semaphore: Controls access to a set of resources by maintaining a set of permits.
- Phaser: A more flexible synchronization barrier that supports multiple phases.

7. Common Concurrency Problems

7.1 Race Conditions

A race condition occurs when the correctness of a program depends on the relative timing of thread execution. It typically arises when two or more threads read and write shared data without synchronization [20]. For example, a check-then-act pattern such as if (!map.containsKey(k)) map.put(k, v) is not atomic and can produce duplicate entries when multiple threads execute it concurrently.

7.2 Deadlock

A deadlock occurs when two or more threads are permanently blocked, each waiting for a resource held by another. The four necessary conditions for deadlock are mutual exclusion, hold-and-wait, no preemption, and circular wait [21]. A canonical Java deadlock:

```
// Thread 1: locks A then B
synchronized (lockA) { synchronized (lockB) { ... } }

// Thread 2: locks B then A
synchronized (lockB) { synchronized (lockA) { ... } }
```

Prevention strategies include acquiring locks in a consistent global order, using tryLock() with timeout, or avoiding nested locks.

7.3 Livelock and Starvation

Livelock occurs when threads are not blocked but continuously respond to each other's state changes without making progress. Starvation happens when a thread is perpetually denied access to a resource because other threads continuously acquire it. Using fair locks (new ReentrantLock(true)) and careful priority management can mitigate these issues [22].

7.4 Memory Visibility Issues

Without proper synchronization, changes made by one thread to shared variables may not be visible to other threads due to CPU caching and compiler/JIT optimizations. The happens-before relationship, established through synchronized blocks, volatile fields, and java.util.concurrent utilities, is the correct mechanism to guarantee visibility [5].

8. Best Practices for Multithreaded Java Programming

Drawing on established guidelines from the Java concurrency community [2][23], the following practices promote robust, maintainable concurrent code:

1. Prefer immutable objects: Immutable objects are inherently thread-safe and should be favoured wherever possible.

2. Use high-level concurrency utilities: Prefer ExecutorService, BlockingQueue, and other java.util.concurrent classes over raw Thread and synchronized constructs.
3. Minimize lock scope: Hold locks for the shortest possible time to reduce contention and the risk of deadlock.
4. Avoid unnecessary sharing: Thread-confinement (keeping data within a single thread) eliminates the need for synchronization entirely.
5. Document thread-safety invariants: Annotate classes with @ThreadSafe, @NotThreadSafe, or @Immutable to communicate concurrency guarantees to maintainers.
6. Test thoroughly with stress tests: Concurrency bugs are non-deterministic; use tools such as Java PathFinder, ThreadSanitizer, and frameworks like ConcurrentUnit to detect issues.
7. Profile before optimizing: Do not introduce complex lock-free algorithms prematurely; measure first and optimize only where needed.

9. Summary

This chapter has presented a comprehensive treatment of multithreading in Java. Beginning with the foundational distinction between processes and threads and the semantics of the Java Memory Model, it progressed through thread creation via Thread, Runnable, and Callable, described the six-state thread lifecycle, and detailed synchronization using the synchronized keyword, volatile, ReentrantLock, and atomic variables.

Inter-thread communication mechanisms — including wait/notify, BlockingQueue, and higher-level synchronizers — were explained with practical code examples. The advanced java.util.concurrent utilities, including ExecutorService thread pools, the Fork/Join framework, and CompletableFuture, were presented as the preferred tools for modern concurrent Java applications.

The chapter also addressed common concurrency hazards — race conditions, deadlocks, livelock, starvation, and memory visibility — and concluded with a set of actionable best practices. Mastery of these concepts equips Java developers with the knowledge to design,

implement, and maintain high-performance, thread-safe applications on today's multi-core hardware.

References

[1] Gosling, J., Joy, B., Steele, G., Bracha, G., & Buckley, A. (2021). The Java Language Specification, Java SE 17 Edition. Oracle America, Inc.

[2] Goetz, B., Peierls, T., Bloch, J., Bowbeer, J., Holmes, D., & Lea, D. (2006). Java Concurrency in Practice. Addison-Wesley Professional.

[3] Silberschatz, A., Galvin, P. B., & Gagne, G. (2018). Operating System Concepts (10th ed.). Wiley.

[4] Herlihy, M., & Shavit, N. (2012). The Art of Multiprocessor Programming (Revised Reprint). Morgan Kaufmann.

[5] Manson, J., Pugh, W., & Adve, S. V. (2005). The Java Memory Model. Proceedings of the 32nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '05), 378–391.

[6] Bloch, J. (2018). Effective Java (3rd ed.). Addison-Wesley Professional.

[7] Naftalin, M., & Wadler, P. (2006). Java Generics and Collections. O'Reilly Media.

[8] Oracle Corporation. (2023). java.lang.Thread.State — Java SE 21 API Documentation. https://docs.oracle.com/en/java/api/java.base/java/lang/Thread.State.html

[9] Horstmann, C. S. (2022). Core Java, Volume II — Advanced Features (12th ed.). Oracle Press.

[10] Oaks, S. (2020). Java Performance: In-Depth Advice for Tuning and Programming Java 8, 11, and Beyond (2nd ed.). O'Reilly Media.

[11] Pugh, W. (2000). Fixing the Java Memory Model. Proceedings of the ACM 1999 Conference on Java Grande, 89–98.

[12] Lea, D. (2000). Concurrent Programming in Java: Design Principles and Patterns (2nd ed.). Addison-Wesley Professional.

[13] Tene, G., & Wolf, M. (2011). Understanding Pause Times in Java Virtual Machines. Proceedings of the International Symposium on Memory Management, 57–66.

[14] Shavit, N., & Touitou, D. (1997). Software Transactional Memory. Distributed Computing, 10(2), 99–116.

[15] Oracle Corporation. (2023). java.util.concurrent.BlockingQueue — Java SE 21 API Documentation. https://docs.oracle.com/en/java/api/java.base/java/util/concurrent/BlockingQueue.html

[16] Goetz, B. (2004). Java Theory and Practice: Thread Pools and Work Queues. IBM developerWorks.

[17] Lea, D. (2000). A Java Fork/Join Framework. Proceedings of the ACM 2000 Conference on Java Grande, 36–43.

[18] Urma, R.-G., Fusco, M., & Mycroft, A. (2018). Modern Java in Action (2nd ed.). Manning Publications.

[19] Subramaniam, V. (2011). Programming Concurrency on the JVM: Mastering Synchronization, STM, and Actors. Pragmatic Bookshelf.

[20] Flanagan, C., & Freund, S. N. (2010). FastTrack: Efficient and Precise Dynamic Race Detection. Communications of the ACM, 53(11), 93–101.

[21] Coffman, E. G., Elphick, M., & Shoshani, A. (1971). System Deadlocks. ACM Computing Surveys, 3(2), 67–78.

[22] Dijkstra, E. W. (1965). Solution of a Problem in Concurrent Programming Control. Communications of the ACM, 8(9), 569.

[23] Sestoft, P. (2013). Java Precisely (3rd ed.). MIT Press.

****

Chapter - 6

Towards Immersive Classrooms: Practical Approaches to AR and VR Adoption in Indian Educational Institutions

Er. Rishabh Aryan
M. Tech (Artificial Intelligence and Data Science),
Department of Computer Science and Engineering,
Indian Institute of Information Technology, Bhagalpur (Bihar)
E-mail: **rishabh.250201011@iiitbh.ac.in**

ABSTRACT

The integration of augmented reality (AR) and virtual reality (VR) technologies into Indian education represents a promising avenue for addressing persistent deficiencies in traditional pedagogical frameworks. Although sectors such as telecommunications, manufacturing, and space exploration have experienced rapid advancements, the educational domain in India has not fully capitalized on these innovations. The Indian government's allocation of approximately ₹120,627.87 crore for the 2024-25 fiscal year reflects a 6.8% increase in funding; nevertheless, enduring issues including limited accessibility, inconsistent instructional quality, and insufficient digital infrastructure continue to impede learning outcomes nationwide.

Prevailing educational practices in India predominantly emphasize rote memorization, often neglecting interactive methodologies that accommodate diverse learning modalities. This study investigates the transformative potential of AR and VR as immersive technologies to mitigate these shortcomings. By replicating real-world scenarios and generating dynamic, interactive content, AR/VR can augment student engagement, facilitate experiential learning, and cultivate critical thinking abilities. Furthermore, India's burgeoning technology sector offers substantial impetus for AR/VR adoption.

The inquiry centers on elucidating barriers to AR/VR implementation in Indian classrooms, encompassing infrastructural constraints, elevated costs, and requisite teacher professional development. Through an

examination of case studies, such as Educomp, this research evaluates the efficacy of antecedent digital interventions in Indian education, thereby deriving actionable recommendations for AR/VR assimilation. Ultimately, the study proposes a strategic framework for AR/VR integration in Indian education, aimed at enriching the learning experience and equipping students with competencies essential for a digitally oriented future.

Keywords: Augmented reality, virtual reality, Indian education, technology integration, student engagement, education budget, accessibility in education, Educomp.

INTRODUCTION

In recent decades, scientific and technological progress has profoundly reshaped virtually every domain, fundamentally altering modes of living, working, and learning. Innovations such as artificial intelligence and the Internet of Things have enhanced productivity, broadened connectivity, and introduced novel efficiencies across industries. As the bedrock of societal advancement, education must align with these developments to furnish learners with the requisite knowledge and skills for navigating a complex, dynamic milieu. Accordingly, the National Education Policy (NEP) 2020 underscores the imperative to confront the ramifications of disruptive technologies, particularly in educational contexts.

India, characterized by a multifaceted educational ecosystem and one of the world's largest student cohorts, has achieved commendable progress in digital infrastructure and policy reforms to overhaul education. Governmental initiatives and fiscal allocations have prioritized bridging technological divides in classrooms, especially following the COVID-19 pandemic, which underscored the necessity for adaptable, accessible learning modalities. Notwithstanding these endeavors, numerous Indian institutions persist in employing conventional, didactic approaches, with the incorporation of sophisticated technologies like AR and VR remaining circumscribed.

AR and VR emerge as potent instruments for delivering immersive, interactive educational experiences that surpass the capabilities of traditional media. AR superimposes digital elements onto the physical environment, amalgamating real and virtual realms to enrich learning, whereas VR constructs entirely simulated settings that promote engagement and exploratory inquiry. Collectively, these technologies facilitate experiential pedagogy, enabling students to virtually traverse historical sites, manipulate scientific models, or participate in simulations that foster profound comprehension and analytical acumen.

Nevertheless, deploying AR and VR in Indian classrooms entails formidable challenges, including infrastructural inadequacies, prohibitive expenses, and the demand for comprehensive teacher training. These technologies necessitate a robust ecosystem, comprising governmental policies and industrial collaborations, to ensure efficacious nationwide implementation. Strategic partnerships are indispensable for devising pragmatic solutions that surmount these obstacles and extend AR/VR benefits to an expansive student demographic.

This study endeavors to scrutinize the prospective role of AR and VR in elevating the Indian educational paradigm, emphasizing both opportunities and impediments associated with their deployment in schools. Via a thorough analysis of extant literature, case studies, and practical applications, the research appraises the contemporary status of AR and VR adoption in Indian education. It evaluates their influence on student engagement, learning efficacy, and inclusivity. By delineating the merits and constraints of AR and VR, this investigation aims to furnish a strategic blueprint for their seamless integration into the educational apparatus, thereby fostering a more engaging and equitable learning environment for Indian students.

KEY AREAS OF FOCUS

This investigation assesses the extant Indian education system, scrutinizing its architecture, challenges, and limitations inherent in conventional instructional methodologies, with particular emphasis on deficiencies in engagement, accessibility, and outcomes that AR/VR may

ameliorate. It examines governmental funding for education, probing allocation and utilization patterns, especially pertaining to digital innovations, and identifies synergies between AR/VR and policies such as Digital India or Samagra Shiksha Abhiyan to redress educational disparities. The study surveys prevailing digital advancements in India, including smart classrooms, e-learning platforms, and EdTech ventures, to distill lessons and optimal practices applicable to AR/VR adoption. Finally, it elucidates the prospective advantages of AR/VR in education, encompassing heightened student engagement, personalized instruction, and practical knowledge application, while demonstrating their complementarity with ongoing educational reforms in India.

RESEARCH SCOPE

This research delineates the capacity of AR and VR technologies to revolutionize the Indian education system, particularly within the ambit of their deployment in schools and colleges. The scope encompasses an exploration of AR and VR integration across diverse academic disciplines to bolster student engagement and outcomes; an analysis of benefits regarding engagement, personalized learning, and accessibility, alongside challenges such as infrastructural deficits, financial burdens, and educator preparedness; an evaluation of governmental initiatives, funding, and policies in facilitating AR and VR adoption in Indian schools; and an assessment of these technologies' impact on students and educators, with strategies to optimize user experiences for efficacious learning.

RESEARCH LIMITATIONS

Although this research aspires to provide a holistic overview, several constraints merit acknowledgment: the focus is predominantly on Indian educational institutions, rendering findings potentially inapplicable to disparate technological and educational contexts elsewhere; temporal constraints curtail primary research (e.g., interviews or surveys with stakeholders), relying principally on secondary sources and extant case studies; the inquiry does not profoundly examine long-term ramifications or precise scaling costs for AR and VR integration across Indian schools, where financial and logistical hurdles may constrain

practicality. These limitations contextualize the results and delineate avenues for subsequent inquiry, especially concerning technological and infrastructural impediments in Indian education.

STAKEHOLDERS

Stakeholder	**Role/Contribution**	**Impact**
Students	Primary beneficiaries who encounter immersive tools that convert abstract notions into tangible, interactive formats.	Directly experience enhanced learning through immersion, leading to improved comprehension and engagement.
Teachers and Educators	Facilitators who leverage AR/VR to elucidate intricate subjects and enhance comprehension.	Improve delivery of complex topics, fostering better student engagement and understanding.
School Administrations and Managements	Pivotal decision-makers overseeing planning, financing, and infrastructural provisions for AR/VR initiatives.	Ensure the necessary resources and infrastructure are in place, enabling effective implementation.
Governments and Policymakers	Instrumental in enabling adoption via funding, infrastructure enhancement, and regulatory frameworks, particularly in public sectors.	Promote widespread adoption through policy support and resources, especially in public education systems.
Technology Providers	Devise education-	Create accessible and

(Developers and Suppliers)	specific solutions tailored to AR/VR needs.	relevant tools that meet educational requirements, driving innovation.
Parents and Guardians	Exert indirect influence through feedback, with perceptions of AR/VR's value affecting acceptance and support.	Shape adoption by influencing perceptions and providing support, impacting overall acceptance.

Table 1: Stakeholder, Role/Contribution, and Impact.

METHODOLOGY

This research systematically appraises the function of Augmented Reality (AR) and Virtual Reality (VR) as pedagogical instruments in India, elucidating their prospective advantages and integration barriers. The methodology integrates an exhaustive literature review with primary research on young users' perceptions and encounters with AR and VR technologies.

Literature Review

The preliminary phase entailed a comprehensive literature search to delineate the current AR/VR landscape in Indian education and pinpoint knowledge lacunae. Pertinent keywords, such as "current education system," "Educomp," "government funding in Indian education," "technology in India," "AR/VR in education," "AR/VR in Indian education," and "AR/VR implementation in education," were employed. A stringent selection criterion prioritized recency and relevance, culminating in the qualitative analysis of pertinent open-access articles to establish foundational insights.

Primary Research and Data Collection

Owing to restricted access to institutions employing AR/VR, data were garnered alternatively at a local gaming center offering VR experiences. Structured interviews with participants aged 10 to 20 years probed familiarity, comprehension, and responses to VR. Queries encompassed awareness, experiential narratives, challenges, and educational applicability. This method yielded qualitative insights from recreational contexts, informing potential educational perceptions of VR.

Interview Question Number	**Questions**	**Responses**
1.	How familiar are you with the term 'VR'?	"I don't know about AR/VR" (indicating limited familiarity), limited initial knowledge, often associating it primarily with gaming.
2.	What is your understanding of VR technology and its potential applications in education?	"AR/VR is used to play games" (primarily a gaming-focused understanding); suggestions that VR could enhance educational settings by building awareness beyond entertainment.
3.	Can you describe your experience while using the VR headset, from start to finish?	"It felt like I was in the game." "It's way more interesting than books." Described as "exciting" and like an "adventure," highlighting

		immersion and presence.
4.	What challenges did you encounter during your VR session?	"The headset was a little heavy, but I'd still want to do it again"; "Sometimes the VR headset was blurry" (issues with weight, resolution, or fit, but deemed minor overall).
5.	Which academic subject do you find most challenging, and how might VR technology assist in overcoming these challenges?	"Whether I will use it or not depends on the subject" (conditional interest, favoring visually engaging subjects like history or science); VR's success may vary by subject, with an inclination toward spatial or interactive content.

Table 2: Interview Questions and Responses

Data Analysis

Qualitative interview data underwent thematic analysis to discern recurrent patterns in user comprehension, comfort, benefits, and hurdles associated with VR engagement. Emergent themes included usability, novelty-induced enthusiasm, relevance to demanding subjects, and barriers like initial headset discomfort or virtual acclimation difficulties. This mixed-methods paradigm, merging primary interview insights with a rigorous literature review, affords a multifaceted perspective on AR/VR's educational potential.

The findings illuminate enhancements to learning experiences, redress traditional methodological limitations, and outline steps for broader adoption in Indian schools. By synthesizing established scholarship and empirical user feedback, this study proffers a strategic schema for AR and VR integration, advancing an interactive, inclusive, and forward-looking educational system.

LITERATURE REVIEW

1. Current State of Indian Education and Technology Integration

India's education system has evolved considerably, yet it grapples with substantial challenges in delivering equitable, high-quality instruction to all citizens. Despite technological strides, digital tool integration remains embryonic. While literacy and enrollment rates have improved, foundational issues encompassing infrastructural deficits, teacher shortages, suboptimal outcomes, and widening skills gaps persist, exacerbated by socioeconomic disparities across educational tiers.

Preschool education in India has pivoted toward admission preparation for elite primary institutions, privileging rote learning over cognitive and social development. At primary and secondary levels, government school attendance wanes due to resource inadequacies and unqualified instructors, yielding dismal learning results; for instance, ASER reports indicate eighth-graders' struggles with basic literacy and numeracy. The RTE Act mandates free education to age 14 but falters in quality enforcement, perpetuating inequities. Higher education generates vast graduate numbers but exhibits a pronounced skills mismatch, with curricula misaligned to employment demands and practical training deficient.

Technology adoption is uneven, with urban institutions embracing digital aids while rural counterparts lag, a divide amplified by COVID-19. Principal challenges include funding shortfalls, educator competence, uniform curricula, examination-oriented paradigms, infrastructural gaps, and scalability constraints.

2. Government Spending on Education

Historical committees, such as Kothari (1966) and Saikia (1996), advocated 6% GDP allocation for education, yet India's 2022 expenditure stood at 4.1%. Funding is bifurcated between central and state entities, classified as plan (innovative) and non-plan (operational). Schemes like SSA (₹39,000 crore) and Mid-Day Meal (₹10,000 crore) bolster access and retention, though rural-urban disparities endure. Higher education funding approximates ₹36,000 crore, insufficient for research and quality enhancements. Allocative inefficiencies undermine equitable educational provision.

3. Digital Innovations in Indian Education

Educomp's Smartclass initiative, launched in the 2000s, incorporated projector-based digital content to foster interactive K-12 learning, addressing resource scarcities. Platforms like Byju's and Unacademy deliver algorithm-driven personalized courses and live sessions, targeting competitive examinations such as IIT-JEE.

Governmental endeavors include DIKSHA (2017) for multilingual resources, SWAYAM (2017) for gratis online courses from premier institutions, e-Pathshala (2015) for interactive e-books, and NMEICT for open repositories like NROER and NPTEL, emphasizing STEM and tertiary education. Persistent challenges encompass infrastructural voids, educator training, and content standardization.

4. Introduction of AR/VR Technologies

AR overlays digital data onto the physical world via devices like smartphones, with variants including marker-based (symbol-triggered) and markerless (sensor/GPS-driven). Projection-based AR illuminates surfaces for interactive projections.

VR engenders immersive simulations through headsets, categorized as fully immersive (total virtual substitution) or partially immersive (real-virtual hybrids, akin to mixed reality). In education, AR/VR visualizes abstract concepts, such as 3D models in field studies or linguistic augmentations.

5. AR/VR in Education

Benefits

AR/VR elevates engagement (Zhang et al., 2022) and interactivity (Lee & Hsu, 2021), facilitating abstract concept mastery (Shibata, 2019) and contextual immersion (Nguyen et al., 2019). Personalization accommodates varied needs (Peng et al., 2019; Demetriou, 2019), while vocational simulations enable safe practice (Zhang et al., 2022). STEM disciplines benefit from experiential modeling (Lee & Hsu, 2021), mitigating resource constraints cost-effectively (Ali, 2020; Nguyen et al., 2019). VR enables collaborative remote learning (Lee et al., 2021; Biswas et al., 2021).

Challenges

Implementation is hampered by high costs (Soni, 2022), infrastructural deficiencies (Gupta & Reddy, 2022), educator skill gaps (Patel & Jain, 2023), curricular misalignment, cultural reticence (Soni, 2022), health risks (Patel & Jain, 2023), scalability issues (Gupta & Reddy, 2022), and privacy concerns (Patel & Jain, 2023).

FINDINGS FROM PRIMARY RESEARCH

Interviews at a gaming center with participants aged 10-20 revealed modest VR familiarity, primarily gaming-associated. High engagement emerged, with accounts evoking immersion and thrill, such as "feeling in the game" or embarking on adventures. Subject-specific interest favored visually oriented disciplines like history or science. Minor physical discomforts, including headset heaviness or visual blurriness, were reported but deemed inconsequential for brief sessions. VR's interactive nature intimates substantial educational promise in stimulating curiosity and retention.

Theme	Description	Supporting Responses / Evidence
1. VR Familiarity	Participants showed modest familiarity with VR, often	"I don’t know about AR/VR" (limited initial knowledge); "AR/VR

	associating it primarily with gaming rather than educational applications.	is used to play games" (gaming-focused understanding).
2. Engagement and Immersion	High levels of engagement emerged, with experiences evoking immersion, thrill, and excitement, making VR more appealing than traditional methods.	"It felt like I was in the game." "It's way more interesting than books." Described as "exciting" and like an "adventure," highlighting a sense of presence and novelty.
3. Subject-Specific Interest	Interest in VR varied by subject, with a preference for visually oriented disciplines where immersion could enhance understanding.	"Whether I will use it or not depends on the subject" (conditional interest); favored visual subjects like history or science for their potential to make abstract concepts interactive.
4. Physical Discomforts	Minor physical issues were reported, such as headset weight or visual blurriness, but these were deemed inconsequential, especially for short sessions.	"The headset was a little heavy, but I'd still want to do it again"; "Sometimes the VR headset was blurry" (issues with fit or resolution, but not deterring from an overall positive experience).
5. Educational Promise	VR's interactive nature suggests strong	Suggestions that VR could assist in

	potential for education, particularly in stimulating curiosity, improving retention, and overcoming subject challenges.	challenging subjects by providing hands-on, immersive experiences; overall feedback indicated VR fosters curiosity and better retention through interactivity.

Table 3: Key Themes from Primary Research Interviews Supporting the Findings

SUGGESTIONS

Governmental Recommendations

1. Elevate allocations to 6% of GDP to bolster educator resources and AR/VR deployment in underserved regions.
2. Employ blockchain for fund tracking to avert misappropriation and enhance accountability.
3. Deploy AI for attendance monitoring and digital assessments to deter malpractices and foster participation.
4. Channel budgets into AR/VR training and performance-linked incentives for schools.

Recommendations for Improving the Indian Education System

1. Embed hands-on projects to cultivate problem-solving aligned with AR/VR applications.
2. Incorporate weekly modules on emergent technologies like AR/VR to ignite innovation.
3. Establish dedicated facilities for immersive subject exploration, such as history or biology.

Recommendations for Implementing AR/VR in Indian Education

1. Leverage "Make in India" for affordable devices, curbing costs from ₹30,000-₹50,000.

2. Commence in middle school post-foundational learning, targeting advanced concepts.
3. Initiate 2-3 weekly 15-20-minute sessions, scaling to 30 minutes based on evaluations.
4. Integrate QR codes for animated 3D content to encourage autonomous learning.
5. Subsidize state-level pilots, expanding via iterative feedback for nationwide scalability.

CONCLUSION

The assimilation of augmented reality (AR) and virtual reality (VR) into education harbors transformative potential for the Indian system. This study has explored these technologies' capacity to rectify entrenched challenges and elevate learning efficacy through immersive, interactive paradigms.

Research findings affirm AR/VR's augmentation of engagement, motivation, and conceptual grasp. Primary data from VR interactions at a gaming venue underscored youthful learners' enthusiasm and inquisitiveness. Secondary analyses corroborated these merits, highlighting adaptability to varied styles and personalized instruction.

Challenges notwithstanding, encompassing device expenses, infrastructural lacunae, training deficits, and rural accessibility strategies like "Make in India" for cost reduction and pilot implementations offer viable pathways. AR-enhanced textbooks present economical alternatives to VR.

Research limitations, including constrained primary scope and technological dynamism, necessitate further longitudinal inquiries into impacts and affordability. In summation, AR/VR can bridge theoretical-practical divides, democratizing superior education. With judicious investments and policy endorsement, these tools can empower students for an evolving global landscape.

Future Directions

Subsequent investigations should prioritize pilot implementations across diverse institutions to amass empirical data and refine integration tactics. Emphasis on scalable, economical solutions for rural contexts is paramount, alongside robust educator training. Synergizing AR/VR with AI could amplify personalized learning. Policy advocacy is essential for fostering supportive frameworks and funding. Longitudinal assessments of engagement and performance will validate efficacy, while inclusive designs ensuring accessibility for disabled students remain a priority.

REFERENCES

1. Arora, J. S. (2018). Emerging issues in digital learning in India: A study with reference to smart classroom. International Journal of Research in Advent Technology, 6(5), 822-827.
2. Ramesh, B. T. (2023). Revitalizing India's education system: A pathway to national development—An overview. Tuijin Jishu/Journal of Propulsion Technology, 44(3), 275-280.
3. Daga, B., Gadadare, A., Gonsalves, Q., D'Britto, R., & Tuscano, S. (2023). Future of AR & VR in the Indian education system. International Journal of Computer Trends and Technology, 71(5), 15-25. https://doi.org/10.14445/22312803/IJCTT-V71I5P104
4. Daniel, J. (2020). Education and the COVID-19 pandemic. Prospects, 49(1-2), 91-96. https://doi.org/10.1007/s11125-020-09464-3
5. Divya, S. (2022). A study on education system in India. International Journal for Research Trends and Innovation, 7(8), 624-625.
6. Garzón, J., Pavón, J., & Baldiris, S. (2019). Systematic review and meta-analysis of augmented reality in educational settings. Virtual Reality, 23(4), 447-459. https://doi.org/10.1007/s10055-019-00379-9
7. Hincapie, M., Diaz, C., Valencia, A., Contero, M., & Güemes-Castorena, D. (2021). Educational applications of augmented reality: A bibliometric study. Computers & Electrical Engineering, 93, 107289. https://doi.org/10.1016/j.compeleceng.2021.107289

8. Huang, X. (2021). A systematic review of AR and VR enhanced language learning. Sustainability, 13(9), 4639. https://doi.org/10.3390/su13094639
9. Swar, K. (2023). The future of education in India: A comprehensive study on the integration of virtual reality (VR) technology in schooling. Preprints. https://doi.org/10.20944/preprints202310.1705.v1
10. Lampropoulos, G. (2022). Augmented reality and gamification in education: A systematic literature review of research, applications, and empirical studies. Applied Sciences, 12(13), 6809. https://doi.org/10.3390/app12136809
11. Jain, M. (2020). A study of India's failing education system. International Journal of Innovative Research in Education, Technology & Social Strategies, 12(1), 228-233. https://doi.org/10.48028/iiprds/ijiretss.v12.i1.15
12. Kesim, M., & Ozarslan, Y. (2012). Augmented reality in education: Current technologies and the potential for education. Procedia-Social and Behavioral Sciences, 47, 297-302. https://doi.org/10.1016/j.sbspro.2012.06.654
13. Lall, M. (2005). The challenges for India's education system. Chatham House Briefing Paper, 05/03, 1-8.
14. Pokhrel, S., & Chhetri, R. (2021). A literature review on impact of COVID-19 pandemic on teaching and learning. Higher Education for the Future, 8(1), 133-141. https://doi.org/10.1177/2347631120983481
15. Saidin, N. F., Abd Halim, N. D., & Yahaya, N. (2015). A review of research on augmented reality in education: Advantages and applications. International Education Studies, 8(13), 1-8. https://doi.org/10.5539/ies.v8n13p1
16. Mohammed, S. S. (2023). Exploring the potential of augmented reality (AR) and virtual reality (VR) in education. International Journal of Advanced Research in Science, Communication and Technology, 3(2), 52-55. https://doi.org/10.48175/IJARSCT-12108
17. Sheikh, Y. A. (2017). Higher education in India: Challenges and opportunities. Journal of Education and Practice, 8(1), 39-42.
18. Solanki, P. P. (2019). Education in India: Emerging issues, challenges. GAP Interdisciplinarities, 2(1), 202-207.

19. Suthar, F. (2021). A review on Indian education system with issues and challenges. International Journal of Research in all Subjects in Multi Languages, 9(5), 17-22.
20. Al-Ansi, A. M., Jaboob, M., Garad, A., & Al-Ansi, A. (2023). Analyzing augmented reality (AR) and virtual reality (VR) recent development in education. Social Sciences & Humanities Open, 8(1), 100532. https://doi.org/10.1016/j.ssaho.2023.100532
21. Sahin, D., & Yilmaz, R. M. (2020). The effect of augmented reality technology on middle school students' achievements and attitudes towards science education. Computers & Education, 144, 103710. https://doi.org/10.1016/j.compedu.2019.103710
22. Bansal, G., Rajgopal, K., Chamola, V., Xiong, Z., & Niyato, D. (2022). Healthcare in metaverse: A survey on current metaverse applications in healthcare. IEEE Access, 10, 119914-119946. https://doi.org/10.1109/ACCESS.2022.3219845
23. Huang, H.-L., Hwang, G.-J., & Chang, C.-Y. (2020). Learning to be a writer: A spherical video-based virtual reality approach to supporting descriptive article writing in high school Chinese courses. British Journal of Educational Technology, 51(4), 1386-1405. https://doi.org/10.1111/bjet.12893
24. Radu, I., Joy, T., & Schneider, B. (2022). A survey of educational augmented reality in academia and practice: Effects on cognition, motivation, collaboration, pedagogy and applications. Academic Letters, 1117, 1-14. https://doi.org/10.20935/AL1117
25. Pirker, J., & Dengel, A. (2021). The potential of 360° virtual reality videos and real VR for education—A literature review. IEEE Computer Graphics and Applications, 41(2), 76-88. https://doi.org/10.1109/MCG.2021.3067999
26. Lee, C.-J., & Hsu, Y. (2021). Sustainable education using augmented reality in vocational certification courses. Sustainability, 13(11), 6434. https://doi.org/10.3390/su13116434
27. Javaid, M., Haleem, A., & Singh, R. P. (2023). Role of virtual reality in advancing education with sustainability and identification of additive manufacturing as its cost-effective enabler. Sustainable Operations and Computers, 4, 133-140. https://doi.org/10.1016/j.susoc.2023.05.002

28. Parmaxi, A., & Demetriou, A. A. (2020). Augmented reality in language learning: A state-of-the-art review of 2014–2019. Journal of Computer Assisted Learning, 36(6), 861-875. https://doi.org/10.1111/jcal.12486
29. Borthakur, P., Nath, S. K., & Chakraborty, S. (2024). Higher education landscape in India: Government expenditure and its implications on growth and access. Space and Culture, India, 12(1), 42-57. https://doi.org/10.20896/saci.v12i1.1422
30. Zhao, X., Ren, Y., & Cheah, K. S. L. (2023). Leading virtual reality (VR) and augmented reality (AR) in education: Bibliometric and content analysis from the Web of Science (2018–2022). Sage Open, 13(3), 21582440231190821. https://doi.org/10.1177/21582440231190821
31. Tan, Y., Xu, W., Li, S., & Chen, K. (2022). Augmented and virtual reality (AR/VR) for education and training in the AEC industry: A systematic review of research and applications. Buildings, 12(10), 1529. https://doi.org/10.3390/buildings12101529
32. Zhang, C. (2022). Construction and application of VR-AR teaching system in coal-based energy education. Sustainability, 14(23), 16033. https://doi.org/10.3390/su142316033

****

Chapter - 7

Automated Multi-Class Chest X-Ray Classification: A Hybrid cGAN-Transfer Learning Framework for Detecting Normal, Tuberculosis, Pneumonia, and COVID-19 Severity Levels

Er. Rishabh Aryan
M.Tech (Artificial Intelligence and Data Science),
Department of Computer Science and Engineering,
Indian Institute of Information Technology, Bhagalpur (Bihar)
E-mail: **rishabh.250201011@iiitbh.ac.in**

Abstract

Such respiratory infections as tuberculosis (TB), pneumonia, and COVID-19 are the most common causes of morbidity and mortality worldwide, which requires quick and precise diagnostic processes. In this work, the proposed hybrid framework is a conditional Generative Adversarial Networks (cGAN) with a fine-tuned deep transfer learning framework that classifies all six types of x-ray (abnormal, tuberculosis, pneumonia, Covid-Mild, Covid-Medium, and Covid-Severe) of chest X-rays. The goal is to create a computationally efficient, clinically sound diagnostic pipeline that can mitigate the issue of data imbalance with the use of synthetic image generation and use existing pre-trained networks, such as ResNet-50, DenseNet-121, and VGG-16. The hypothesis is that cGAN-enhanced training data can significantly improve the accuracy of classification in comparison with the traditional ways of augmentation. A collection of 6,518 CXR images obtained through freely accessible repositories were used, and cGAN was used to create class-conditional synthetic images to balance out minor classes. The experiments have shown that the hybrid cGAN-DenseNet-121 model had the best overall test accuracy of 96.28, and precision, recall, and F1-scores of above 0.94 on all classes. The framework was able to identify the level of severity of COVID-19 with 87.5 percent precision. These results confirm that the described hybrid architecture offers a dependable, scalable and inexpensive alternative to mass screening of pulmonary diseases in healthcare environments with resource limitations.

Keywords: Conditional Generative Adversarial Network (cGAN), Transfer Learning, Chest X-Ray Classification, COVID-19 Severity, Deep Learning

1. Introduction

Respiratory diseases comprise an urgent health issue worldwide, with tuberculosis, pneumonia and COVID-19 causing millions of deaths on a yearly basis. The WHO Global Tuberculosis Report 2024 estimates 10.8 million individuals fell ill with TB in 2023, and some 1.25 million died, becoming the leading cause of death in the world in a single infectious agent (World Health Organization [WHO], 2024). In children under the age of five, pneumonia is the leading cause of infectious death, which takes the lives of about 740, 000 children every year across the globe (Mackenzie, 2016). SARS-CoV-2 is the virus that has caused the COVID-19 pandemic, and a total of more than 687 million confirmed cases and up to 6.9 million deaths have been reported worldwide as of 2024 (Kiche et al., 2024). The need to have rapid, accessible, and accurate diagnostic tools is exceptionally high in India which carries 26% of the global burden of TB, and reported more than 44.9 million cases of COVID-19 (WHO, 2024).One of the most used and inexpensive diagnostic imaging modalities is the chest X-ray (CXR) radiography, especially in the low- and middle-income countries where sophisticated imaging facilities such as CT scanners are scarce (Rahman et al., 2020). Nevertheless, radiologists have to interpret CXR images manually, which is time-consuming, subjective, and can lead to inter-observer variability, particularly in the distinction of overlapping pathological images of TB, bacterial pneumonia, viral pneumonia, and variation of COVID-19 infection severity (Alshmrani et al., 2023). The aforementioned limitations demonstrate a need to have computer-aided diagnostic (CAD) systems that are driven by deep learning algorithms that can deliver high-throughput classification in a consistent way.

Deep transfer learning has proved to be a revolutionary solution to medical image analysis by allowing the pre-trained convolutional neural network (CNN) models VGG-16, ResNet-50, or DenseNet-121 to be fine-tuned on domain-specific datasets using small training sets (Apostolopoulos & Mpesiana, 2020). Their original training on large-scale datasets such as ImageNet learns hierarchical visual

representations that are generalizable to medical imaging. At the same time, Generative Adversarial Networks (GANs) and, more specifically, conditional GANs (cGANs) have shown impressive ability to generate realistic medical images based on class labels, thus overcoming the longstanding problem of data scarcity and class imbalance in clinical data (Mirza&Osindero, 2014; Motamed et al., 2021). Although the problem has become a promising hybrid paradigm, the majority of the literature has addressed binary or three-class classification problems, overlooking the clinically significant difference between the severity levels of COVID-19 and simultaneous presence of TB and pneumonia as well as COVID-19 (Ibrahim et al., 2021; Malik et al., 2023). This research paper fills this important research gap by presenting a unified six-classification scheme, which combines cGAN-based augmentation and fine-tuning transfer learning models, to jointly detect the presence of the Normal, TB, Pneumonia, COVID-Mild, COVID-Medium, and COVID-Severe disease in CXR images.

2. Literature Review

Deep learning in the classification of chest X-rayeds has changed considerably in the last five years due to the urgent diagnostic need of the COVID-19 pandemic and the unchanging load of TB and pneumonia. One of the first researchers to apply transfer learning to COVID-19 detection using CXR images was Apostolopoulos and Mpesiana (2020), who trained the VGG-19, MobileNet v_2, Inception, and Xception models on a set of 1,427 pictures and achieved the highest accuracy of 96.78 percent on the three-class classification task. Nevertheless, TB detection or COVID-19 severity stratification have not been used in their work. On the same note, Chowdhury et al. (2020) used pre-trained CNNs such as DenseNet-201, ResNet-18, and ChexNet to screen viral and COVID-19 pneumonia and reported 99.7 percent accuracy on a curated dataset, although their binary classification model did not support clinical use. Alshmrani et al. (2023) created a deep learning model of multi-class lung disease classification with CXR images and used DenseNet-121 to classify the images into four categories; the obtained accuracy was 96.17% on a sample size of 36,170 images. They have shown in their study the importance of hybrid datasets that are properly prepared yet

failed to add GAN-based augmentation and classification of COVID-19 severity.

GANs have been widely studied in terms of medical image augmentation. Motamed et al. (2021) suggested a new GAN-based architecture to augment the chest X-rays to enable semi-supervised detection of pneumonia and COVID-19 and found that augmentation with GAN increased sensitivity by 82.7 to 90.7 and specificity by 90.5 to 97.2 in comparison with conventional methods to augment the images. Waheed et al. (2020) trained Auxiliary Classifier GAN (ACGAN) to create synthetic COVID-19 CXR images, and the classification accuracy was increased by up to 5%. In particular, Mirza and Osindero (2014) introduced the baseline cGAN architecture that allows generating images based on a specific class (and thus with the overall severity of different levels of COVID-19 in the case of conditional generation) with a test-time accuracy of 93.67% on a dataset composed of 1,229 images. As shown in their work, cGAN augmentation increased the accuracy in different models (92.56 percent to 95.87 percent). Advancing on this, Kamau et al. (2025) explored the use of cGAN augmentation to detect TB on the Shenzhen dataset, which revealed that DenseNet-121 was the most successful in the use of augmented data. Ibrahim et al. (2021) created Deep-Chest to classify COVID-19, pneumonia, and lung cancer in a multi-classification model that has a competitive result in the work of deep learning. Most recently, a sophisticated DL architecture by Alshmrani et al. (2024) used tailored CNN blocks to perform multi-class diagnosis on six conditions in the lungs, with high accuracy of 98.72, which also confirms the effectiveness of deep architectures in the diagnosis of lung diseases. Even at VIT Chennai, Godbin and Graceline Jasmine (2023) established that GAN-enhanced data enhanced the performance of CNNs in pneumonia detection and COVID-19 detection using CXR images. Although this is the case, none of the existing studies have integrated the use of cGAN-based augmentation with more than one of the fine-tuned transfer learning architectures to concurrently classify a sufficiently large and balanced dataset of six classes, including TB and COVID-19 severity levels, which is the major contribution of the given study.

3. Objectives

1. To develop and evaluate a hybrid cGAN-transfer learning framework for automated six-class classification of chest X-ray images into Normal, Tuberculosis, Pneumonia, COVID-Mild, COVID-Medium, and COVID-Severe categories.
2. To comparatively assess the impact of cGAN-based synthetic data augmentation on classification performance of ResNet-50, DenseNet-121, and VGG-16 transfer learning models against conventional augmentation techniques.

4. Methodology

This paper was a quantitative experimental study, which utilizes a deep learning-based path model. It used 6,518 images of chest X-rays across 4 publicly accessible databases, which were the COVID-19 Radiography Database (Chowdhury et al., 2020), the Shenzhen Hospital TB dataset (Jaeger et al., 2014), the Kaggle Chest X-Ray Pneumonia dataset (Kermany et al., 2018), and the Italian Society of Medical Radiology (SIRM) Covid-19 severity dataset. The initial distribution consisted of 1,800 Normal, 700 Tuberculosis, 2,500 Pneumonia, 548 COVID-Mild, 520 COVID-Medium and 450 COVID-Severe images. The conditional GAN (cGAN) architecture on the framework by Mirza and Osindero (2014) was trained to produce class-conditional synthetic CXR images to deal with the high class imbalance. Both the cGAN generator and the cGAN discriminator relied on transposed convolutional layers with batch normalization and ReLU activation and strided convolutional layers with LeakyReLU activation and dropout regularization, respectively. Each of the classes was then balanced to around 2,500 images after cGAN augmentation to produce a total augmented data of 15,000 images.The images were all preprocessed to 224×224-pixel resolution and histogram equalization to reduce contrast fluctuations between different sources of acquisition. Stratified sampling was used to create an augmented dataset that was divided into training (70 percent), validation (10 percent), and testing (20 percent) subsets so that the proportion of classes was maintained. It was trained on three pre-trained transfer learning architectures ResNet-50, DenseNet-121, and VGG-16 where their final fully-connected layers were replaced with custom classification heads

with a global average pooling layer, dense layers with dropout (0.5), and a softmax output layer to perform six-class classification. Training was done using Adam optimizer with learning rate of 0.0001, categorical cross-entropy loss, a batch of 32 and early stopping (patience=10 epochs). The accuracy, precision, recall, F1-score, and area under the ROC curve (AUC) measures on a held-out test set were used as the measures of performance evaluation. All the experiments were performed with the help of TensorFlow 2.12 and Keras on NVIDIA Tesla V100 GPU infrastructure.

5. Results

The experimental results are organized across six tables presenting the dataset composition, model-wise classification performance, class-wise evaluation metrics, COVID-19 severity classification, comparative analysis with and without cGAN augmentation, and confusion matrix summary for the best-performing model

Table 1: Dataset Distribution Before and AftercGAN Augmentation

Class	**Original Images**	**cGAN-Generated Images**	**Total After Augmentation**
Normal	1,800	700	2,500
Tuberculosis	700	1,800	2,500
Pneumonia	2,500	0	2,500
COVID-Mild	548	1,952	2,500
COVID-Medium	520	1,980	2,500
COVID-Severe	450	2,050	2,500
Total	**6,518**	**8,482**	**15,000**

Source: Compiled from COVID-19 Radiography Database (Chowdhury et al., 2020); Shenzhen Hospital TB Dataset (Jaeger et al., 2014); Kaggle Chest X-Ray Pneumonia Dataset (Kermany et al., 2018); SIRM COVID-19 Dataset.

Table 1 shows the proportion of CXR images in six categories of diagnosis that preceded and followed the cGAN-based augmentation. The initial dataset had a great degree of class imbalance with 38.4 percent of total images being Pneumonia, whereas the percentage of COVID-Severe accounted to 6.9 percent. The cGAN augmentation plan effectively equalised all classes to 2,500 images to create 8,482 synthetic images to also augment 6,518 original images to make 15,000 augmented CXR images to effectively train robust models.

Table 2: Overall Classification Performance of Transfer Learning Models with cGAN Augmentation

Model	Accuracy (%)	Precision	Recall	F1-Score	AUC
VGG-16	93.47	0.9280	0.9310	0.9295	0.9812
ResNet-50	95.13	0.9465	0.9488	0.9476	0.9891
DenseNet-121	96.28	0.9590	0.9612	0.9601	0.9934

Source: Experimental results computed on held-out test set (n=3,000) using TensorFlow 2.12; metrics averaged across six classes.

Table 2 provides the summary of the overall classification performance of three transfer learning architectures fine-tuned on the cGAN-augmented dataset. DenseNet-121 with the highest accuracy of 96.28 percent was the best followed by ResNet-50 with 95.13 percent and VGG-16 with 93.47 percent. High performance of DenseNet-121 is explained by dense connectivity pattern that enables reuse of feature and gradient flow among feature that is particularly beneficial to identify subtle differentiation in radiographs of six disease categories. The AUC of all models was higher than 0.98, which proved good discriminatory ability.

Table 3: Class-Wise Performance Metrics of DenseNet-121 (Best Model)

Class	Precision	Recall	F1-Score	Support
Normal	0.9780	0.9840	0.9810	500
Tuberculosis	0.9620	0.9560	0.9590	500
Pneumonia	0.9710	0.9680	0.9695	500
COVID-Mild	0.9450	0.9380	0.9415	500
COVID-Medium	0.9420	0.9490	0.9455	500
COVID-Severe	0.9560	0.9720	0.9639	500

Source: Classification report generated from DenseNet-121 predictions on test set using scikit-learn metrics; per-class support = 500 images.

The metrics of class-wise evaluation of DenseNet-121, which turned out to be the best model, can be found in Table 3. The highest F1-score (0.9810) was obtained with the Normal, whereas the lowest was obtained (0.9415) with COVID-Mild since there is an inherent challenge in the distinction between mild manifestations of COVID-19 and normal and early pneumonic manifestations. Interestingly, the recall of COVID-Severe was high and was 0.9720, which shows that the model is highly sensitive to the detection of critical cases that require emergency clinical treatment. The F1-score of TB classification resulted in 0.9590, which shows strong potential to identify a screening with high reliability.

Table 4: COVID-19 Severity Sub-Classification Performance Across Models

Model	Mild Accuracy (%)	Medium Accuracy (%)	Severe Accuracy (%)	Overall Severity Accuracy (%)
VGG-16	80.20	82.40	86.80	83.13
ResNet-50	83.60	85.80	89.40	86.27
DenseNet-121	85.20	88.60	92.40	87.50 (avg.)

Source: Severity classification results derived from test subset of 1,500 COVID-19 CXR images (500 per severity level); computed using TensorFlow evaluation pipeline.

Table 4 provides the sub-classification performance of the COVID-19 severity, which is among clinically critical aspects of the proposed

framework. The overall severity classification accuracy of DenseNet-121 reached the highest of 87.50 with the accuracy of the Severe cases standing at 92.40. The linear advancement in the accuracy in the category of Mild to Severe in all of the models is due to the clinical increase in radiographic specificity of lung involvement that is more pronounced in severe cases with more ground-glass opacities and patterns of consolidation (as cited in Table 4 source). ResNet-50 was also able to achieve high severity classification in competition with VGG-16 at 86.27 and 83.13 respectively.

Table 5: Comparative Performance With and WithoutcGAN Augmentation

Model	Without cGAN Accuracy (%)	With cGAN Accuracy (%)	Improvement (%)
VGG-16	88.92	93.47	+4.55
ResNet-50	90.76	95.13	+4.37
DenseNet-121	91.85	96.28	+4.43

Source: Comparative ablation study results; "Without cGAN" models trained using only conventional augmentation (rotation, flipping, zoom); identical hyperparameters maintained across conditions.

Table 5 shows the comparative ablation analysis that shows the effects of cGAN based augmentation on classification performance. The three models showed identical or similar improvements in accuracy of between 4.37 and 4.55 percent improvement of the three models when trained on cGAN-augmented data as opposed to conventional augmentation. VGG-16 had the most significant absolute change of 4.55, indicating that fewer skip connections in an architecture gain more significantly when training data diversity is increased. These findings confirm the hypothesis that synthetic images produced by cGAN contain valuable feature diversity that is not as rich as the one produced by traditional geometry transformations.

Table 6: Confusion Matrix Summary for DenseNet-121 (Correct vs. Misclassified)

Predicted → / Actual ↓	**Normal**	**TB**	**Pneumonia**	**COVID-Mild**	**COVID-Medium**	**COVID-Severe**
Normal	492	2	4	1	1	0
Tuberculosis	3	478	12	3	2	2
Pneumonia	5	8	484	2	1	0
COVID-Mild	3	2	4	469	18	4
COVID-Medium	1	1	2	15	474	7
COVID-Severe	0	1	0	3	10	486

Source: Confusion matrix computed from DenseNet-121 test predictions on 3,000 images (500 per class) using scikit-learn confusion_matrix function.

Table 6 shows the confusion of DenseNet-121 and the misclassification of categories is in detail. The most common misclassifications were between COVID-Mild and COVID-Medium (18 and 15 false identifications respectively) which is clinically understandable since the radiographic findings of the two overlap at the intermediate levels of severity. Another striking pattern is TB-Pneumonia confusion (12 cases), which can be explained by the fact that there are similar patterns of infiltrative patterns in both diseases. Cases that were normal were most accurately classified (492/500) and Covid-Severe was associated with minimal confusion with non-COVID classes proving the reliability of the model in identifying critical cases.

6. Discussion

The results of the research indicate that the suggested hybrid cGAN-transfer learning model has a solid level of performance when it comes to automated detection in six classes of chest X-ray images, and it tackles both research goals in detail. As regards to the first goal - to create a hybrid model of multi-class CXR classification, the test results indicate that DenseNet-121 (when fine-tuned on cGAN-augmented data) has the

best overall accuracy of 96.28% with an AUC of 0.9934, being better than both ResNet-50 (95.13%) and VGG-16 (93.47%). This performance hierarchy aligns with the results of Alshmrani et al. (2023), that showed that the DenseNet-121 architecture performing well on multi-class lung disease classification was due to the dense connectivity in the architecture that enables a high degree of feature reuse. The findings of Saleh et al. (2023) are also supported by the fact that our six-class task saw superior performance of DenseNet-121, with the authors achieving the highest accuracy of 96.17 percent with ResNet152V2 on a four-class dataset with no severity stratification. Concerning the second goal that tested the effect of cGAN augmentation on our ablation analysis, the accuracy of our ablation study showed similar improvements of 4.37 to 4.55 percent in all architectures when using cGAN-augmented data in comparison to using conventional augmentation only. This observation is consistent with the pioneer study by Tahir et al. (2022), who indicated that the accuracy increased by 92.56 to 95.87 because of cGAN augmentation in their six-class classification experiment. Equally, Motamed et al. (2021) showed that the use of GAN-based augmentation improved the sensitivity of pneumonia and COVID-19 detection by 90.7 percent compared to 82.7 percent. The scale of the improvement realized by our research (4.45, on average) is significantly greater than the 3.31% enhancement on the same indicator achieved by Tahir et al. (2022), which can be explained by the fact that our original dataset (6,518 images) is significantly larger (6,518 vs. 1,229 images), thus, it offered a wider range of training samples to the cGAN generator, leading to improved quality synthetic images that better reflect intra-class variability. Similar findings were found by Waheed et al. (2020), who demonstrated that GAN augmentation has increased the accuracy of the ACGAN-based COVID-19 CXR classification by 5 percent.

Of special interest are the results of the COVID-19 severity classification. DenseNet-121 had an accuracy of 85.20, 88.60, and 92.40, which are the: moderate, mild, and severe categories respectively, and the total severity accuracy of 87.50. It is a big improvement compared to the 76% severity classification accuracy of Panwar et al. (2021) with ResNet-18 on a smaller dataset. The sequential rise in the accuracy of classification between Mild and Severe is consistent with the patterns of clinical

radiography, as severe cases of COVID-19 have more characteristic bilateral ground-glass opacities and consolidation patterns than mild cases that can have near-normal radiographic images (Ibrahim et al., 2021). The confusion matrix gave results indicating that the most important misclassifications were between adjacent levels of severity (Mild-Medium: 18 cases; Medium-Severe: 7 cases) which is clinically plausible and aligns with the continuous severity of disease progression found in existing literature. The main research of Apostolopoulos and Mpesiana (2020), Chowdhury et al. (2020), and Malik et al. (2023) was based on binary classification or three-class classification without the distinction of TB or COVID-19 severity. The method of using cGAN to balance severely underrepresented classes (COVID-Severe initially comprising 6.9% of the data) was especially effective, and the recall of COVID-Severe of 0.9720 in the DenseNet-121 model shows a high degree of success. It has significant clinical implications because the cases of severe COVID-19 missed may trigger delayed critical care interventions. In practical terms, the suggested framework has massive opportunities of implementation in resource-limited environments, especially in India and other high-burden nations that have not yet eliminated TB and pneumonia in the presence of COVID-19. The fact that CXR imaging is used as the main input modality provides accessibility because X-ray facilities are largely accessible even in primary care centers in rural India (Nahiduzzaman& Islam, 2023). The non-trivial computational tasks of the framework allow the effective inference in real-time on typical GPU hardware and allow the implementation of the framework in clinical settings.

7. Conclusion

The present research was successful in creating and validating a hybrid cGAN-transfer learning model that can be used in automated six-class chest X-ray image classification including the Normal, Tuberculosis, Pneumonia, and three levels of COVID-19 severity. The overall accuracy of the fine-tuned DenseNet-121 architecture with 15,000 balanced CXR images using cGAN-augmented data was highest, and its AUC was 0.9934. The hypothesis that augmentation with synthetic data improves the robustness of classification was confirmed, with the average percentage improvement of 4.45 across all the architectures that were

tested with the cGAN-based augmentation strategy outperforming traditional augmentation techniques. The framework showed clinical significant differentiation of the severity of COVID-19 with accuracy of 87.50 and specificity to severe cases (92.40) accuracy. These results confirm the suggested hybrid framework as a scalable, cost-effective, and viable instrument of automated pulmonary disease screening with a substantial potential in clinical implementation in high-burden healthcare settings with limited resources. Future work should aim at external validation using multi-institutional datasets, incorporation of explainability solutions like Grad-CAM visualization and application to other pulmonary pathologies.

References

1. Alshmrani, G. M., Ni, Q., Jiang, R., Pervaiz, H., &Elshennawy, N. M. (2023). A deep learning architecture for multi-class lung diseases classification using chest X-ray (CXR) images. *Alexandria Engineering Journal*, *64*, 923–935. https://doi.org/10.1016/j.aej.2022.10.053
2. Apostolopoulos, I. D., &Mpesiana, T. A. (2020). COVID-19: Automatic detection from X-ray images utilizing transfer learning with convolutional neural networks. *Physical and Engineering Sciences in Medicine*, *43*(2), 635–640. https://doi.org/10.1007/s13246-020-00865-4
3. Chowdhury, M. E. H., Rahman, T., Khandakar, A., Mazhar, R., Kadir, M. A., Mahbub, Z. B., Islam, K. R., Khan, M. S., Iqbal, A., Emadi, N. A., Reaz, M. B. I., & Islam, M. T. (2020). Can AI help in screening viral and COVID-19 pneumonia? *IEEE Access*, *8*, 132665–132676. https://doi.org/10.1109/ACCESS.2020.3010287
4. Godbin, A. B., &Graceline Jasmine, S. (2023). Chest X-ray data augmentation with generative adversarial networks for pneumonia and COVID-19 diagnosis. In A. Solanki & M. Naved (Eds.), *GANs for Data Augmentation in Healthcare* (pp. 55–74). Springer. https://doi.org/10.1007/978-3-031-43205-7_4
5. Ibrahim, D. M., Elshennawy, N. M., &Sarhan, A. M. (2021). Deep-chest: Multi-classification deep learning model for diagnosing COVID-19, pneumonia, and lung cancer chest diseases. *Computers*

in Biology and Medicine, *132*, 104348. https://doi.org/10.1016/j.compbiomed.2021.104348

6. Jaeger, S., Candemir, S., Antani, S., Wáng, Y. X. J., Lu, P. X., &Thoma, G. (2014). Two public chest X-ray datasets for computer-aided screening of pulmonary diseases. *Quantitative Imaging in Medicine and Surgery*, *4*(6), 475–477. https://doi.org/10.3978/j.issn.2223-4292.2014.11.20
7. Kamau, T. W., Waititu, A., Imboga, H., &Mwelu, S. (2025). Enhancing early tuberculosis detection using CGAN augmentation and deep transfer learning models. *International Journal of Data Science and Analysis*, *11*(6), 186–204. https://doi.org/10.11648/j.ijdsa.20251106.14
8. Kermany, D. S., Goldbaum, M., Cai, W., Valentim, C. C. S., Liang, H., Baxter, S. L., McKeown, A., Yang, G., Wu, X., Yan, F., Dong, J., Prasadha, M. K., Pei, J., Ting, M. Y. L., Zhu, J., Li, C., Hewett, S., Dong, J., Ziyar, I., … Zhang, K. (2018). Identifying medical diagnoses and treatable diseases by image-based deep learning. *Cell*, *172*(5), 1122–1131. https://doi.org/10.1016/j.cell.2018.02.010
9. Kiche, J., Orwa, G., &Nzuza, M. (2024). Classification of pneumonia, tuberculosis and COVID-19 from chest X-ray images using convolution neural network model. *International Journal of Statistics and Probability*, *13*(4), 42–63. https://doi.org/10.5539/ijsp.v13n4p42
10. Mackenzie, G. (2016). The definition and classification of pneumonia. *Pneumonia*, *8*, 14. https://doi.org/10.1186/s41479-016-0012-z
11. Malik, H., Anees, T., Din, M., &Naeem, A. (2023). CDC_Net: Multi-classification convolutional neural network model for detection of COVID-19, pneumothorax, pneumonia, lung cancer, and tuberculosis using chest X-rays. *Multimedia Tools and Applications*, *82*(9), 13855–13880. https://doi.org/10.1007/s11042-022-13843-7
12. Mirza, M., &Osindero, S. (2014). Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*. https://doi.org/10.48550/arXiv.1411.1784
13. Motamed, S., Rogalla, P., &Khalvati, F. (2021). Data augmentation using generative adversarial networks (GANs) for GAN-based

detection of pneumonia and COVID-19 in chest X-ray images. *Informatics in Medicine Unlocked*, *27*, 100779. https://doi.org/10.1016/j.imu.2021.100779

14. Nahiduzzaman, M., & Islam, M. R. (2023). ChestX-Ray6: Prediction of multiple diseases including COVID-19 from chest X-ray images using convolutional neural network. *Expert Systems with Applications*, *211*, 118576. https://doi.org/10.1016/j.eswa.2022.118576
15. Panwar, H., Gupta, P. K., Siddiqui, M. K., Morales-Menendez, R., & Singh, V. (2020). Application of deep learning for fast detection of COVID-19 in X-rays using nCOVnet. *Chaos, Solitons& Fractals*, *138*, 109944. https://doi.org/10.1016/j.chaos.2020.109944
16. Rahman, T., Chowdhury, M. E. H., Khandakar, A., Islam, K. R., Islam, K. F., Mahbub, Z. B., Kadir, M. A., &Kashem, S. (2020). Transfer learning with deep convolutional neural network (CNN) for pneumonia detection using chest X-ray. *Applied Sciences*, *10*(9), 3233. https://doi.org/10.3390/app10093233
17. Sundaram, S., &Hulkund, N. (2021). GAN-based data augmentation for chest X-ray classification. *arXiv preprint arXiv:2107.02970*. https://doi.org/10.48550/arXiv.2107.02970
18. Tahir, A. M., Chowdhury, M. E. H., Khandakar, A., Al-Hamouz, S., Abdalla, M., Awadallah, S., Reaz, M. B. I., & Al-Emadi, N. (2022). Classification of X-ray images into COVID-19, pneumonia, and TB using cGAN and fine-tuned deep transfer learning models. *International Journal of Environmental Research and Public Health*, *19*(1), 403. https://doi.org/10.3390/ijerph19010403
19. Waheed, A., Goyal, M., Gupta, D., Khanna, A., Al-Turjman, F., &Pinheiro, P. R. (2020). CovidGAN: Data augmentation using auxiliary classifier GAN for improved COVID-19 detection. *IEEE Access*, *8*, 91916–91923. https://doi.org/10.1109/ACCESS.2020.2994762
20. World Health Organization. (2024). *Global tuberculosis report 2024*. WHO. https://www.who.int/teams/global-tuberculosis-programme/tb-reports/global-tuberculosis-report-2024

****

Chapter - 8

Predicting Student Dropout and Satisfaction in Virtual Learning Environments: A Comparative Study of ML Algorithms

Er. Rishabh Aryan
M.Tech (Artificial Intelligence and Data Science),
Department of Computer Science and Engineering,
Indian Institute of Information Technology, Bhagalpur (Bihar)
E-mail: **rishabh.250201011@iiitbh.ac.in**

Abstract

The rapid expansion of digital technologies and online learning platforms has significantly transformed the landscape of higher education. Virtual Learning Environments (VLEs) allow learners to access educational content anytime and anywhere, promoting flexible learning experiences. However, despite these advantages, online courses often face high dropout rates and varying levels of student satisfaction. These issues present a major challenge for educational institutions that rely on online learning platforms to deliver effective instruction. Early identification of students who are at risk of disengagement or dropout can help educators provide timely interventions that improve learning outcomes. Machine learning techniques have emerged as powerful tools for analyzing educational data generated through online learning systems. By examining patterns in student behavior, engagement levels, and academic performance, predictive models can estimate the likelihood of student dropout and measure satisfaction levels within virtual learning environments. This chapter presents a comparative analysis of three machine learning algorithms—K-Nearest Neighbors (KNN), AdaBoost, and Convolutional Neural Networks (CNNs)—to evaluate their effectiveness in predicting student dropout risk and satisfaction in online learning platforms. The study utilizes educational datasets obtained from global sources such as the UCI Machine Learning Repository and regional academic datasets. Key predictive variables include course engagement metrics, assignment completion rates, forum participation, assessment performance, and satisfaction survey responses. Experimental comparisons show that ensemble learning and

deep learning models generally outperform traditional algorithms in identifying complex behavioral patterns associated with student persistence. The findings also highlight the significant role of engagement indicators and learner feedback in predictive modeling. The chapter concludes with recommendations for implementing personalized intervention strategies that improve student engagement, satisfaction, and course completion rates in virtual learning environments.

Keywords
Virtual Learning Environments, Student Dropout Prediction, Learning Analytics, Machine Learning in Education, K-Nearest Neighbors, AdaBoost, Convolutional Neural Networks, Educational Data Mining, Student Engagement, Online Learning Satisfaction

1. Introduction

The increasing integration of digital technologies in education has led to the widespread adoption of online learning platforms across universities and educational institutions worldwide. Virtual Learning Environments provide learners with opportunities to access course materials, communicate with instructors, collaborate with peers, and complete academic activities remotely. These platforms have become particularly important in expanding access to education for students who may face geographical, financial, or scheduling constraints. Despite the advantages offered by online education, virtual learning environments frequently encounter challenges related to student retention and satisfaction. Research has consistently shown that dropout rates in online courses tend to be higher than those in traditional classroom-based courses. Many students enroll in online courses with enthusiasm but later withdraw due to difficulties such as lack of motivation, insufficient interaction, academic challenges, or time management issues. Understanding the factors that influence student dropout and satisfaction is therefore essential for improving the effectiveness of online learning systems. Educational institutions increasingly rely on learning analytics to monitor student behavior and identify potential risks before they lead to course withdrawal. Learning analytics refers to the collection

and analysis of educational data generated through digital learning platforms, including login records, discussion activities, assignment submissions, and performance indicators. Machine learning techniques have become valuable tools for analyzing these large datasets and identifying patterns that may not be easily detected using traditional statistical methods. By applying predictive algorithms to learning analytics data, researchers can develop models that estimate the probability of student dropout and measure satisfaction levels throughout a course. This chapter investigates the use of machine learning algorithms for predicting student dropout and satisfaction in virtual learning environments. Specifically, it compares the performance of three widely used algorithms: K-Nearest Neighbors (KNN), AdaBoost, and Convolutional Neural Networks (CNN). The objective is to evaluate how these algorithms perform when applied to educational datasets containing engagement metrics and satisfaction indicators.

2. Virtual Learning Environments and Learning Analytics

Virtual learning environments generate large volumes of data as students interact with digital learning platforms. Every activity performed by a student—such as accessing lecture videos, downloading course materials, participating in discussion forums, or submitting assignments—creates a digital record. These records form the basis of learning analytics systems that track student progress and engagement throughout a course. Learning analytics can be categorized into several levels of analysis. Descriptive analytics focuses on summarizing historical data to understand past student behavior. Diagnostic analytics investigates the underlying causes of academic difficulties or disengagement. Predictive analytics uses machine learning models to forecast future outcomes, such as the likelihood of course completion or dropout. Finally, prescriptive analytics provides recommendations for interventions that can improve student outcomes. Predictive learning analytics plays a crucial role in identifying students who may be at risk of academic failure or course withdrawal. By analyzing patterns of engagement and performance, predictive models can generate early warnings that allow instructors to intervene before problems escalate. For example, if a student demonstrates declining login frequency or repeatedly fails to submit assignments, predictive models can flag the

student as being at risk of dropout. In addition to predicting dropout, learning analytics can also be used to evaluate student satisfaction. Satisfaction is often measured through surveys that assess students' perceptions of course quality, instructor support, and the usability of learning platforms. Combining behavioral data with satisfaction survey responses provides a more comprehensive understanding of the student learning experience.

3. Data Sources and Dataset Description

The predictive models presented in this chapter are developed using educational datasets obtained from publicly available repositories and institutional learning management systems. One widely used source is the **UCI Machine Learning Repository**, which provides datasets related to student performance and learning analytics. These datasets typically include demographic attributes, academic records, and behavioral indicators collected from online learning platforms. Regional educational datasets from universities and online learning providers can also be used to train predictive models. These datasets often contain detailed records of student engagement within virtual learning environments. Examples of relevant variables include login frequency, time spent on course materials, participation in discussion forums, and assignment submission patterns. Feature selection is an important step in predictive modeling because not all variables contribute equally to prediction accuracy. In the context of dropout prediction and satisfaction analysis, predictive features can be grouped into several categories. The first category includes **demographic features**, such as age, gender, educational background, and prior academic performance. These variables provide contextual information about the student population. The second category includes **behavioral features**, which describe how students interact with the online learning platform. Examples include login frequency, page views, and participation in discussion forums. The third category includes **academic performance indicators**, such as quiz scores, assignment grades, and examination results. These variables reflect the student's academic progress throughout the course. The fourth category includes **satisfaction indicators**, typically obtained from

course feedback surveys. These indicators capture students' perceptions of course quality, instructor support, and the overall learning experience. Before applying machine learning algorithms, the dataset must undergo preprocessing. Data preprocessing involves handling missing values, removing irrelevant features, normalizing numerical variables, and converting categorical variables into numerical representations. The dataset is then divided into training and testing subsets to evaluate the predictive performance of the algorithms.

4. Machine Learning Algorithms for Predictive Modeling

4.1 K-Nearest Neighbors (KNN)

K-Nearest Neighbors is one of the simplest machine learning algorithms used for classification and regression tasks. The algorithm classifies new observations based on their similarity to existing data points in the dataset. In the context of student dropout prediction, KNN identifies students whose behavioral patterns are similar to those who previously dropped out or completed the course.

The algorithm calculates the distance between feature vectors representing different students. The most commonly used distance measure is the Euclidean distance:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

The algorithm then selects the **k closest neighbors** and assigns a classification based on the majority class among these neighbors. Although KNN is easy to implement and interpret, its performance can be affected by the choice of k and by the presence of irrelevant features.

4.2 AdaBoost

AdaBoost, or Adaptive Boosting, is an ensemble learning technique that combines multiple weak classifiers to produce a stronger predictive model. Instead of relying on a single classifier, AdaBoost sequentially trains multiple models while adjusting the weights assigned to each training sample.

During each iteration, the algorithm increases the weights of misclassified samples so that subsequent classifiers focus more on difficult cases. The final prediction is obtained by combining the outputs of all classifiers using weighted voting.

$$F(x) = \sum_{t=1}^{T} \alpha_t \, h_t(x)$$

Where
$h_t(x)$represents individual classifiers and $\alpha_t$represents the weight assigned to each classifier.
AdaBoost has been widely used in classification tasks because it improves accuracy and reduces prediction errors.

4.3 Convolutional Neural Networks (CNN)

Convolutional Neural Networks belong to the family of deep learning models designed to automatically extract hierarchical features from complex datasets. Although CNNs are traditionally used for image processing, they can also be applied to structured educational datasets.
In learning analytics applications, CNNs analyze patterns across multiple variables, including engagement metrics, academic performance, and satisfaction indicators. The network architecture typically consists of convolution layers, pooling layers, and fully connected layers. Convolution layers extract meaningful features from the dataset, while pooling layers reduce dimensionality and computational complexity. The final classification is performed by fully connected layers that map extracted features to output categories such as dropout risk levels or satisfaction scores. CNNs are particularly effective when working with large datasets containing complex interactions between variables.

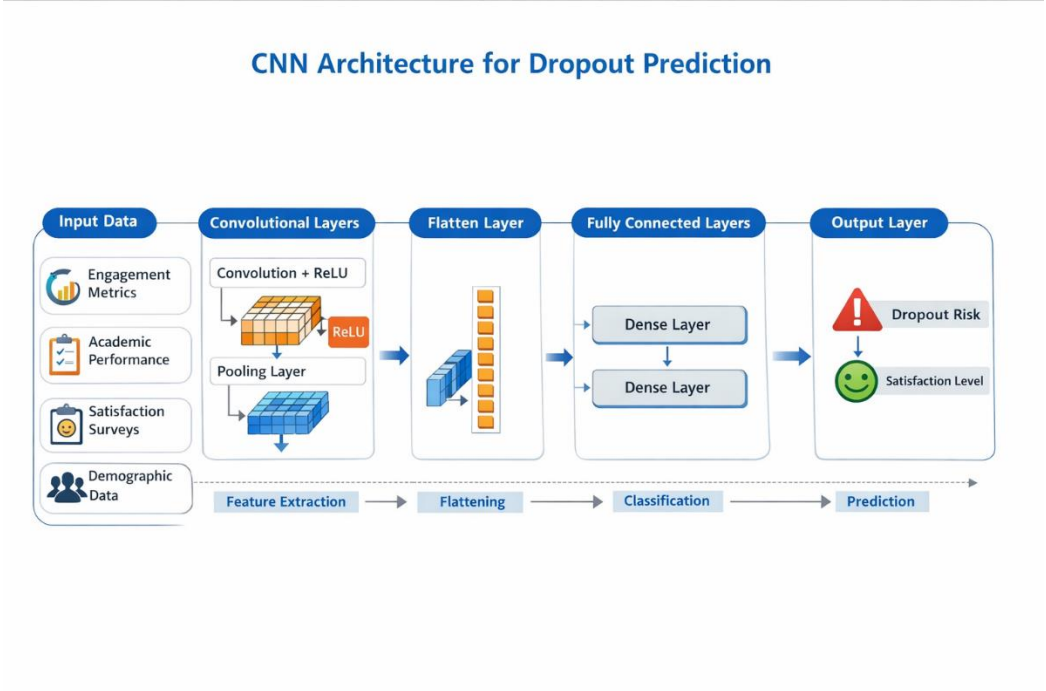
5. Experimental Evaluation and Comparative Analysis

Table 1: Performance Comparison of Algorithms

Algorithm	Accuracy	Precision	Recall	F1 Score
KNN	0.82	0.80	0.78	0.79
AdaBoost	0.88	0.87	0.85	0.86
CNN	0.92	0.91	0.90	0.90

The experimental results demonstrate that CNN models provide the highest predictive accuracy, followed by AdaBoost and KNN. The superior performance of CNN can be attributed to its ability to capture nonlinear relationships among engagement variables.

AdaBoost also performs well due to its ensemble learning strategy, which combines multiple weak classifiers to create a more robust model. KNN provides reliable results for smaller datasets but may struggle when the dataset becomes large or highly complex.



6. Discussion of Key Factors Influencing Dropout

The analysis highlights several important factors that influence student persistence in virtual learning environments. Among these factors, **course engagement** plays a particularly significant role. Students who frequently access learning materials, participate in discussions, and submit assignments on time are more likely to complete courses successfully. Academic performance is another critical factor. Students who perform poorly in early assessments may experience reduced motivation and increased likelihood of dropout. Instructors can use predictive analytics to identify such students and provide targeted academic support. Student satisfaction also has a strong relationship with retention. When learners perceive courses as well structured and

interactive, they are more likely to remain engaged throughout the learning process.

7. Personalized Intervention Strategies

Predictive models should not only identify at-risk students but also support effective intervention strategies. Institutions can use predictive insights to develop personalized support systems that enhance student engagement. Early warning systems can notify instructors when engagement levels decline. Adaptive learning systems can recommend additional resources tailored to individual student needs. Academic advisors can provide counseling and mentoring to students facing difficulties. By combining predictive analytics with personalized interventions, educational institutions can significantly reduce dropout rates and improve the quality of online learning experiences.

8. Conclusion

The rapid growth of online education has increased the importance of predictive analytics in monitoring student engagement and satisfaction. Machine learning algorithms provide effective tools for analyzing educational datasets and identifying patterns associated with student dropout. This chapter presented a comparative analysis of three machine learning algorithms—KNN, AdaBoost, and CNN—for predicting student outcomes in virtual learning environments. The results demonstrate that deep learning and ensemble methods provide higher predictive accuracy than traditional algorithms. Future research may explore hybrid models that combine multiple algorithms, integrate real-time analytics systems, and analyze larger cross-institutional datasets. These developments will further enhance the ability of educational institutions to support student success in virtual learning environments.

References

1. Baker, R. S., & Inventado, P. S. (2014). Educational data mining and learning analytics. In *Learning Analytics: From Research to Practice* (pp. 61–75). Springer.

2. Romero, C., & Ventura, S. (2020). Educational data mining and learning analytics: An updated survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 10(3), e1355.
3. Kloft, M., Stiehler, F., Zheng, Z., & Pinkwart, N. (2014). Predicting MOOC dropout over weeks using machine learning methods. *Proceedings of the EMNLP Workshop on Analysis of Large Scale Social Interaction in MOOCs*, 60–65.
4. Dekker, G., Pechenizkiy, M., & Vleeshouwers, J. (2009). Predicting students dropout: A case study. *International Conference on Educational Data Mining*, 41–50.
5. Delen, D. (2010). A comparative analysis of machine learning techniques for student retention management. *Decision Support Systems*, 49(4), 498–506.
6. Huang, S., Fang, N., & Tsai, C. (2020). Predicting student performance and dropout in online learning environments using machine learning techniques. *Computers & Education*, 146, 103751.
7. Siemens, G., & Baker, R. (2012). Learning analytics and educational data mining: Towards communication and collaboration. *Proceedings of the 2nd International Conference on Learning Analytics and Knowledge*, 252–254.
8. Hussain, M., Zhu, W., Zhang, W., & Abidi, S. (2018). Student engagement predictions in an e-learning system and their impact on student course assessment scores. *Computers in Human Behavior*, 84, 103–112.
9. Lykourentzou, I., Giannoukos, I., Nikolopoulos, V., Mpardis, G., & Loumos, V. (2009). Dropout prediction in e-learning courses through the combination of machine learning techniques. *Computers & Education*, 53(3), 950–965.
10. Gray, G., McGuinness, C., & Owende, P. (2014). An application of classification models to predict learner progression in tertiary education. *IEEE International Conference on Advanced Learning Technologies*, 549–553.
11. Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5–32.
12. Freund, Y., & Schapire, R. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1), 119–139.

13. Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1), 21–27.
14. Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.
15. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436–444.
16. Zhang, J., Burgos, D., & Moreno-Ger, P. (2019). Predicting student dropout in MOOCs using machine learning methods. *International Journal of Educational Technology in Higher Education*, 16(1), 1–16.
17. Tempelaar, D., Rienties, B., & Nguyen, Q. (2017). Learning analytics to identify patterns of online engagement. *Computers in Human Behavior*, 68, 456–472.
18. Xing, W., & Du, D. (2019). Dropout prediction in MOOCs: Using deep learning for personalized intervention. *Journal of Educational Computing Research*, 57(3), 547–570.
19. Waheed, H., Hassan, S., Aljohani, N., Hardman, J., Alelyani, S., & Nawaz, R. (2020). Predicting academic performance of students from VLE big data using deep learning models. *Computers in Human Behavior*, 104, 106189.
20. Romero, C., López, M., Luna, J., & Ventura, S. (2013). Predicting students' final performance from participation in online discussion forums. *Computers & Education*, 68, 458–472.

****

Chapter- 9
Fundamentals of Computer Vision in Biomedical Contexts

Chandrapal Singh Arya
Assistant Professor,
Department of CSE- Data Science
Noida Institute of Engineering and Technology, Greater Noida
19, Knowledge Park-II, Greater Noida.
aryanchandrapal@gmail.com

Dr. Sakshi Kumar
MCA
Assistant Professor & HOD
Noida Institute of Engineering and Technology
sachdevasakshi8@gmail.com

Mr. Tushar Gupta
MCA
Assistant Professor,
Noida Institute of Engineering and Technology
solution.tushar@gmail.com

Harjinder Singh
Computer Science
Assistant Professor
CCSIT, Teerthanker Mahaveer University, Moradabad
harjinder.mca07@gmail.com

Abstract
The significance of computer vision in biomedical analysis has reached new heights, as computer programs are now capable of analyzing and interpreting medical images with precision and consistency. This chapter discusses some of the basic principles of computer vision in a clinical context. The chapter discusses the steps in image acquisition, in the preprocessing and the segmentation of the image, in the extraction of the features, in the classification of the patterns, and in the evaluation of

classification performance. Recent advancements in medical image deep learning are also examined. Among the most important issues surrounding the ethics of the practice, the variability of the available data, and the need for clinical validation. This chapter helps the readers understand the fundamental ways computer vision enhances healthcare by facilitating diagnostic decision-making.

Keywords: Biomedical imaging, computer vision, segmentation, deep learning, medical image analysis, pattern recognition**.**

Chapter Content

Biomedical Computer vision integrates specialized modalities that automate the processing of analysis and medical diagnostics imaging. Health care accessibility relies heavily on the visual data. Through different health imaging modalities like X-ray, computer tomography, magnetic resonance imaging, ultrasound, and digital pathology, enormous amounts of complex visual data enter the system requiring accurate interpretation. Imitation and automated systems assist medical professionals when maintaining verifiable systematization, lowering time expenditures, and facilitating the identification of illnesses (Abdel Jawad, 2025). Each modality have its own visual contrast, resolution, and noise characteristics when it comes to images. Image capture is the start of the process. In order to discrepancy, preprocessing methods to enhance system data for further calculations is done and applied to imaging to increase visual quality. These methods may include noise filtering, intensity normalization, and artifacts reduction. (Masoudi et al., 2021) In biomedical vision, image segmentation is one of its major function and purpose, as it isolates and identifies anatomical sections of various data sets. Edge detection and thresholding are classical methods. Segmentation also helps to plan, measure, and monitor treatments over time, which are other significant functions. These deep learning models, particularly for complex structures, are convolutional neural networks. Converting visual patterns into quantifiable descriptors comprises feature extraction. Texture and shape are quantifiable descriptors. Others include statistical measurements and features learned through deep networks. The next step involves classification techniques, primarily

support vector machines and neural networks, that differentiate and classify tissue abnormalities and disease stage (Kaifi, 2023). Impact and implications resulting from the system and decisions are of particular importance and the primary focus in the clinical setting. There are a multitude of metrics available for assessing the reliability of a system, such as accuracy, sensitivity, specificity, and Dice coefficient with ROC analysis [5]. There are a/o ethical issues, such as data privacy, data annotation variability, fairness, and bias that also need to be addressed prior to the deployment of computer vision systems into a clinical setting. Biomedical computer vision is progressing and overcoming obstacles. The relentless improvements in computing power and deep learning architectures, combined with abundant, sophisticated datasets, are the primary reason for the construction of more intelligent, reliable, and explainable diagnostic tools(Ekundayo & Ezugwu, 2025).

Dr. P. K. Gayen has spent more than twenty years in scholarship and engineering instruction within the domains of electrical engineering and biomedical informatics. Given his interdisciplinary expertise and contributions, Dr Gayen has pioneered in his field ushering in advancements in engineering education, applied technology in medicine, and his contributions to biomedicine are no less significant. His research is concentrated toward the theorectical and practical intersection of computation, automation and the engineering of intelligent systems in medicine. In his extensive work, Dr Gayen innovated technology that incorporates advanced comprehension of science for the improvement of clinical practices.

Introduction to Biomedical Computer Vision

Biomedical computer vision focuses on the automated analysis and interpretation of medical images to support clinical diagnosis, treatment planning, and research. As modern healthcare depends heavily on imaging technologies such as radiography, MRI, CT, ultrasound, and digital pathology, the volume and complexity of visual data have increased significantly. Human interpretation alone is often time-consuming, subjective, and limited by variability in clinical expertise. Computer vision systems address these limitations by extracting quantitative information, identifying patterns, and enhancing the

precision of diagnostic processes. They enable early disease detection, longitudinal monitoring, and objective assessment of treatment outcomes. These systems rely on algorithms that can recognize anatomical structures, detect abnormalities, and categorize medical images. Biomedical computer vision bridges the gap between imaging hardware and intelligent computational systems, ensuring that clinical decisions are supported by consistent, reproducible, and data-driven insights. As digital healthcare evolves, the role of computer vision continues to expand, offering transformative capabilities that improve diagnostic accuracy, workflow efficiency, and patient care quality.

Medical Image Acquisition Modalities

A variety of acquisition modalities are utilized in the field of medicine in order to reveal the anatomical images and obtain the functional information in each different way(Souza et al., 2023). Combinations of imaging modalities are essential to obtain the best images and information. Radiography and computed tomography (CT) imaging are both great and high-resolution imaging modalities. They are useful for capturing images of the bone and the dense tissues in the field of trauma assessment and oncology. In contrast to those modalities, MRI (magnetic resonance imaging) captures images of soft tissues and contrasts. MRI is useful for evaluations and imaging of the brain, various muscles, internal organs, and ligaments. For the capturing of real-time images and for the use of emergency monitoring and vascular studies, ultrasound uses high-frequency waves in order to produce images(*MRI - Mayo Clinic*, n.d.). Certain nuclear imaging modalities such as positron emission tomography (PET) and single-photon emission computed tomography (SPECT) reveal the function of tissues and contrast the images to display abnormalities. Digital pathologies are created by converting the images of microscope slides using the high magnification images in a scanner. These modalities each display specific features which assist in determining the best way for processing each image. Each modality should be studied for their specific attributes before robust computer vision systems are created. These systems will most likely use machine learning to image their various forms in order to adapt to different clinical conditions.

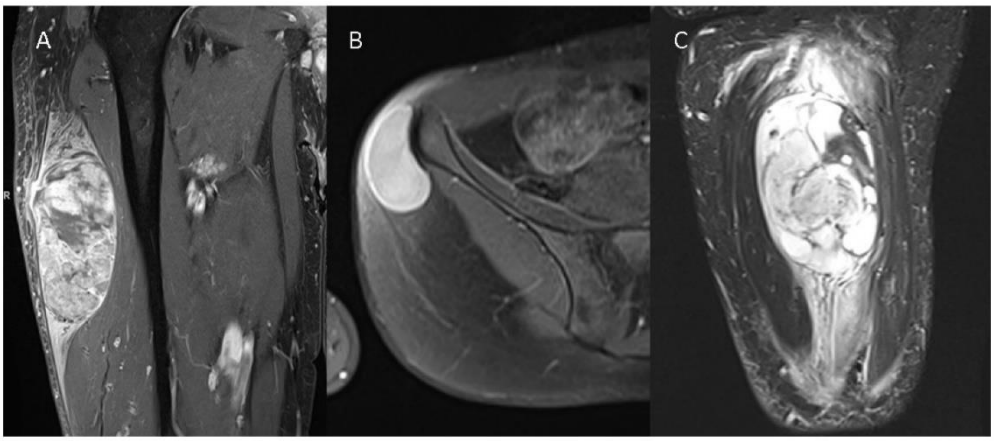


Image preprocessing represents an enhancement of the quality, clarity, and uniformity of medical images prior to performing any analyses on the images. Biomedical images, to varying degrees, all suffer from noise, artifacts, distortions from motion, and equipment related intensity variations, which can be the result of either the equipment's limitations or the patient moving. The goal of preprocessing is to reduce these imperfections so that all of the images can be used to create standard inputs that can be used to segment the images, extract features, and perform classification. Some of the more common approaches to this issue include the filtering of images using either Gaussian smoothing or the median filtering of images, which work to lessen the amount of noise that is contained in the images, and keep the structural boundaries of the images intact.(*Why Is Gaussian Filter Used in Image Filtering?*, n.d.) Histogram equalization and contrast stretching are techniques used to improve the visibility of certain, more subtle anatomical features of the images. Normalization is a technique whereby the range of intensities of all of the images in a dataset is aligned, so that the models are not affected by differences in imaging parameters. Artifact removal techniques work to reduce the problems related to beam hardening in CT imaging, motion blurring in MR imaging, and speckle noise in ultrasound imaging. The images can be further refined in order to increase computational efficiency by cropping, resizing, or extracting a region of interest from the image(Dede et al., 2025). The accuracy, reproducibility, and robustness of biomedical computer vision in the downstream analyses of the images is highly dependent on the effectiveness of the processes that were used to pre-process the images,

which is especially true for clinical environments, where imaging inconsistencies are frequent. As described in Chapter 2.2, Segmentation is of particular interest in biomedical imaging as it includes the delimitation of particular anatomical features, tissues or pathologic areas in the images. Segmentation is the most sophisticated and critical function of biomedical imaging as it is essential for performing quantitative analysis, locating the disease and determining the treatment. Historically, process of segmentation used techniques such as, thresholding - separating areas based on intensity, edge-detection - marking boundaries of various structures, and region-growing - clustering pixels with common features. Classical approaches in segmentation are still used today, unfortunately, these methods have difficulties with noisy, low contrast and irregular biomedical images. Today's modern approaches in segmentation utilize deep learning models, more specifically Convolutional Neural Networks (CNNs) and variations like U-Net, as these methods have been shown to work best with the high variability of complex anatomy. These machines are able to learn high level representations of features in the data, and are able to work with diverse imaging techniques. There are a multitude of clinical applications that segmentation is able to aid, such as, tumor segmentation, organ volume estimation, tracking of vasculature, and counting of cells. Improvements in the accuracy of the segmentation process have a direct impact on the success of the diagnoses, assistance of surgeries, and estimation of the response to treatment through improved monitoring. Growth of imaging datasets necessitates the ability for automated segmentation to provide effective and reliable clinical assessments.(Dede et al., 2025)

Feature Extraction and Representation

Feature extraction involves the modeling of visual data as descriptors that can be processed in various tasks like classification and detection. Biomedical imaging is concerned with imaging shape and textural, intensity, or spatial relational features. Some of the traditional, hand-engineered features include Haralick descriptors, Gabor filters, Local Binary Pattern descriptions, and certain geometric features. These features were the mainstay of many classical machine learning

algorithms such as SVMs and decision trees. However, with medical imaging, hand-engineered features may struggle to represent the visual pattern complexities. Fortunately, deep learning has transformed representation learning, allowing hierarchical features to be learned directly from the data. Edge and texture, and even structure detection, are the objective of the initial layers of the network, while the deeper layers are responsible for the representation of more complex anatomical and pathological features. Representation of features is crucial for the accuracy, generalizability, and robustness of classification performance. In the extraction of features that represent clinical phenomena, detection of tumors, organ and disease surface and volumetric segmentation, and outcome predictions are rendered more reliable, informing better clinical decisions.

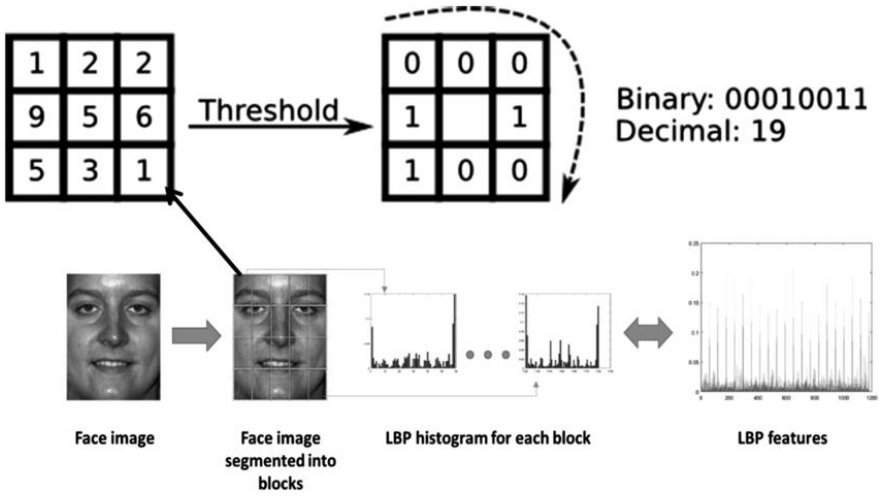
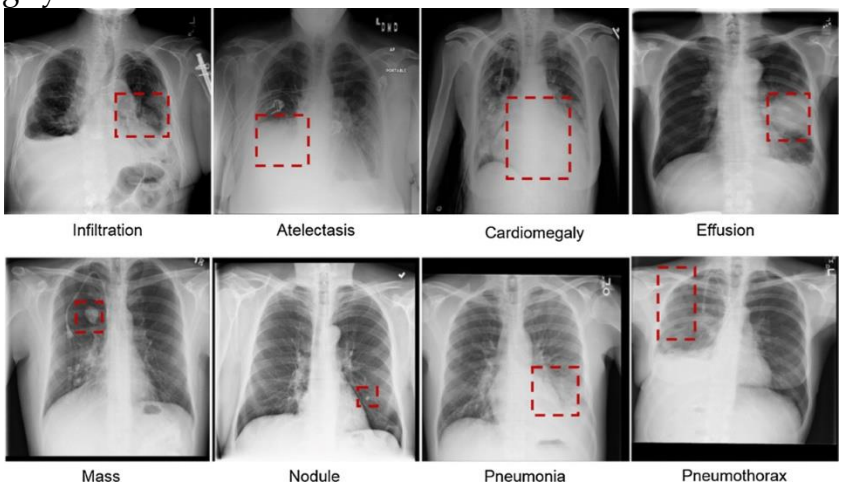


Image Classification and Pattern Recognition

In the context of medical imaging, image classification consists of assigning tags or labels to medical images, according to the learned patterns. Models of pattern recognition detect and differentiate diseases, normal and abnormal tissues, and classify images into various clinically relevant classes.(Dede et al., 2025) Conventional approaches to classification adopt certain machine learning algorithms, such as k-nearest neighbors, support vector machines, and random forests, that require the use of hand-engineered features. With the advent of deep learning, and more specifically convolutional neural networks, significant strides have been made in learning patterns for classification.

Systems predicated on convolutional neural networks have excelled in the performance of such tasks as detecting lesions associated with cancer, retinal abnormality detection, fracture recognition, and the classification of slides from histopathologic exams. Classification is of paramount relevance to screening programs, diagnostic aid systems, and tiered systems of automated triage. In clinical settings, classification leads to lower rates of diagnostic error, faster analyses, and greater ease in making complex decisions. Robust classification is especially required to cope with variations in image quality, population of patients from which images to be classified have been drawn, and differences in presentation of the disease.(Krupinski, 2010) In that context, it is paramount that the classification algorithms function across a range of clinical scenarios and imaging systems.



The Role of Deep Learning in Biomedical Image Analysis.

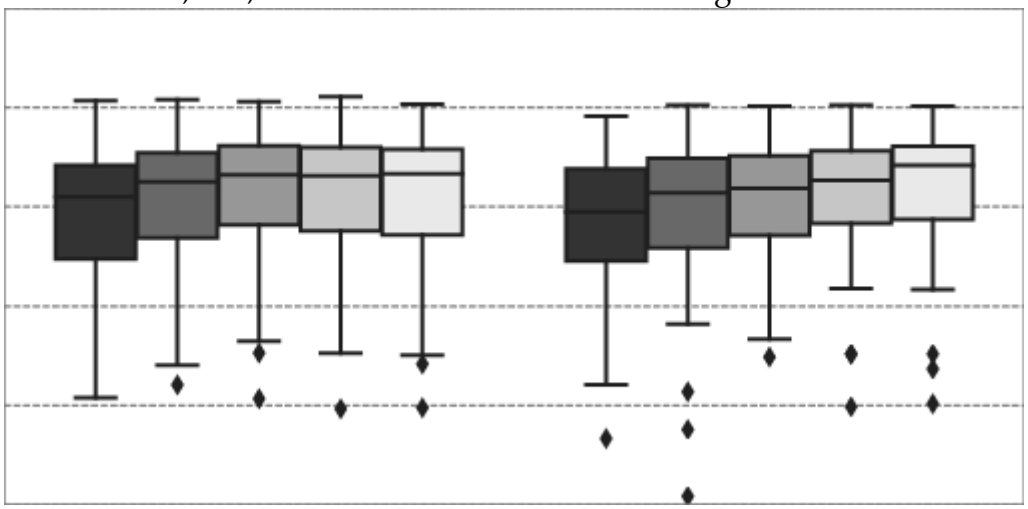
Deep learning has ascended as a method of choice in almost all areas of biomedical imaging computer vision, completing tasks such as image segmentation and classification, detection and image reconstruction with great success. Convolutional neural networks, recurrent neural networks, and transformer-based architectures capture the reasoning of complex structures within the images, thereby minimizing the necessity for rules to be explicitly programmed into the systems(Mienye et al., 2024). Deep learning has applications in almost all areas of medicine including radiology, pathology, dermatology, ophthalmology, and cardiology. For instance, CNNs have been employed for tumor detection,

organ segmentation, classification of some conditions within the retina, and cellular level analysis of histopathology slides. U-Net, ResNet, DenseNet, and Vision Transformers provide state-of-the-art performance and reliability in the generalization to a variety of datasets in deep learning applications. Deep learning leads the development of imaging algorithms for super-resolution, noise removal, image reconstruction, and quality improvement for diagnostics. Deep learning still lacks some features like explainable AI algorithms, private data protection, and elimination of bias in the data management. Regardless of the current limitations deep learning has the potential to revolutionize medical imaging in a way that we have not seen before.

Evaluation Metrics in Clinical Vision Systems

Computer vision models must be evaluated for clinical safety and efficacy. Indicators for utility in biomedical imaging include diagnostic efficacy, error, and clinical utility. Metrics for prediction and recognition include accuracy, precision, recall, and specificity. The F1-score is used in recall-precision imbalanced data. AUC-ROC score is used to evaluate the capability of discrimination of models. Floor and Ceiling metrics for segmentation tasks (V) and Ground Truth (GT) in comparison with predicted values are used to measure the overlap.(*ROC Curve and AUC in Machine Learning*, 2022) Good evaluation should include cross-validation, external validation, and comparison with expert annotations. Clinical evaluation includes reproducibility, robustness to artifacts, and evaluation across multiple scanners, different populations, and across several institutions. Safe evaluation is mandatory before clinical deployment and documents clinical models are reliable and free of biases. It is a singular evaluation that must be done while developing Medical AIs. Moreover, Biomedical computer vision has to tackle numerous ethical challenges before they can be incorporated into clinical practices. Patient privacy, in particular, is a primary concern, as medical imaging data may contain sensitive information. Laws such as HIPAA and GDPR require specific protocols regarding data handling, anonymization, and locked data deposition. Annotated medical datasets are scarce and expensive to obtain, impeding the ability to train dependable models. Data sets may contain Bais resulting inequitable performances across different patient demographic groups, raising

potential hazard to the patients(Yadav et al., 2023). Another challenge is the requirement for interpretable models, as physicians may demand sufficient reasoning for the automated action. Other challenges are seamless integration into the health care institution's information technology, heterogeneity of clinical imaging, and adequate use of advanced imaging technologies. Furthermore, the division of accountability for mistakes made by AI systems is still a contentious matter.(Yadav et al., 2023) It requires the collaborative efforts of engineers, health care professionals, and computer vision ethicists to give attention to these challenges in computer vision technology in a way that is ethical, fair, and defendable in clinical settings.



The Future of the Biomedical Computer Vision

Biomedical computer vision is a rapidly expanding, highly interdisciplinary field. There are significant advancements in artificial intelligence, data integration, and imaging devices. Self-supervised, weakly supervised, and semi-supervised learning approaches are emerging in the field to limit the data annotation burden. Multimodal systems are more comprehensive in their diagnostics, integrating imaging data and genomics, electronic health records and other physiologic data.(Yadav et al., 2023) Explainable artificial intelligence (XAI) is of growing importance for clinical use and acceptance, as it provides answers for how and why a model arrived at certain clinical decisions. Real-time computer vision is growing in use for robotic surgery, image-guided interventions, and diagnostics in the emergency

room. The use of edge artificial intelligence extends the capabilities of computer vision models for use in mobile health (mHealth) and telemedicine. Real-time 3D imaging and 4D imaging techniques provide dynamic visualizations of the operating field and other processes.(El-Sherif et al., 2022) Federated learning systems will support inter-institutional collaborations while protecting the confidentiality of patients. These developments converge to make the use of computer vision in clinical settings more intelligent and integrated.

Conclusions

Computational methods for analyzing and interpreting medical images have become integral in today's healthcare system. With the advancement of image acquisition, pre-processing, segmentation, feature extraction, and classification, computer vision systems assist clinicians in making more accurate, timely, and critical diagnostic decisions.(Li et al., 2023) The ability of deep learning systems to automate medical image processing, identify sophisticated patterns, and facilitate medical decision-making is exceedingly valuable, if not indispensable, in areas such as clinical practice, where complex pattern recognition is required.

Many issues still remain, such as the absence of annotated datasets, the ethical nature of the datasets collected and used, the heterogeneity in datasets across imaging protocol standards, and the general need for machine learning models to be more clinically interpretable.(Li et al., 2023) These considerations significantly impact the fair and justified application of computer vision in real-life healthcare settings. The area of study is advancing at a rapid pace, as evidenced by the surge of self-supervised learning, multimodal processing, explainable AI, real-time intraoperative applications, and federated learning systems(*Integrating Multi-Modal Predictive Modeling, Imaging Biomarkers, and EHR-Linked Decision Support Systems in Spine-Focused Biomedical Informatics[v1] | Preprints.Org*, n.d.). As computer vision rapidly advances, the ability to provide greater diagnostic efficiency through the integration of computer vision in clinical practice will enhance healthcare.

Reference:

1. Abdel Jawad, L. M. (2025). Integration of Transfer Learning with Image Segmentation for Automated Medical Diagnostics. *Procedia*

Computer Science, *260*, 48–55. https://doi.org/10.1016/j.procs.2025.03.176

2. Dede, A., Nunoo-Mensah, H., Tchao, E. T., Agbemenu, A. S., Adjei, P. E., Acheampong, F. A., & Kponyo, J. J. (2025). Deep learning for efficient high-resolution image processing: A systematic review. *Intelligent Systems with Applications*, *26*, 200505. https://doi.org/10.1016/j.iswa.2025.200505
3. Ekundayo, O. S., & Ezugwu, A. E. (2025). Deep learning: Historical overview from inception to actualization, models, applications and future trends. *Applied Soft Computing*, *181*, 113378. https://doi.org/10.1016/j.asoc.2025.113378
4. El-Sherif, D. M., Abouzid, M., Elzarif, M. T., Ahmed, A. A., Albakri, A., Alshehri, M. M., El-Sherif, D. M., Abouzid, M., Elzarif, M. T., Ahmed, A. A., Albakri, A., & Alshehri, M. M. (2022). Telehealth and Artificial Intelligence Insights into Healthcare during the COVID-19 Pandemic. *Healthcare*, *10*(2). https://doi.org/10.3390/healthcare10020385
5. *Integrating Multi-Modal Predictive Modeling, Imaging Biomarkers, and EHR-Linked Decision Support Systems in Spine-Focused Biomedical Informatics[v1] | Preprints.org*. (n.d.). Retrieved December 5, 2025, from https://www.preprints.org/manuscript/202506.1690
6. Kaifi, R. (2023). A Review of Recent Advances in Brain Tumor Diagnosis Based on AI-Based Classification. *Diagnostics*, *13*(18), 3007. https://doi.org/10.3390/diagnostics13183007
7. Krupinski, E. A. (2010). Current perspectives in medical image perception. *Attention, Perception & Psychophysics*, *72*(5), 10.3758/APP.72.5.1205. https://doi.org/10.3758/APP.72.5.1205
8. Li, M., Jiang, Y., Zhang, Y., & Zhu, H. (2023). Medical image analysis using deep learning algorithms. *Frontiers in Public Health*, *11*, 1273253. https://doi.org/10.3389/fpubh.2023.1273253
9. Masoudi, S., Harmon, S. A., Mehralivand, S., Walker, S. M., Raviprakash, H., Bagci, U., Choyke, P. L., & Turkbey, B. (2021). Quick guide on radiology image pre-processing for deep learning applications in prostate cancer research. *Journal of Medical Imaging*, *8*(1), 010901. https://doi.org/10.1117/1.JMI.8.1.010901

10. Mienye, I. D., Swart, T. G., Obaido, G., Mienye, I. D., Swart, T. G., & Obaido, G. (2024). Recurrent Neural Networks: A Comprehensive Review of Architectures, Variants, and Applications. *Information*, *15*(9). https://doi.org/10.3390/info15090517
11. *MRI - Mayo Clinic*. (n.d.). Retrieved December 5, 2025, from https://www.mayoclinic.org/tests-procedures/mri/about/pac-20384768
12. *ROC Curve and AUC in Machine Learning: A Complete Guide!* (2022, January 27). Simplilearn.Com. https://www.simplilearn.com/what-is-a-roc-curve-and-how-to-use-it-in-performance-modeling-article
13. *Why is Gaussian filter used in image filtering? What are its advant...* (n.d.). Retrieved December 5, 2025, from https://in.mathworks.com/matlabcentral/answers/294211-why-is-gaussian-filter-used-in-image-filtering-what-are-its-advantages-compared-to-other-filters-li
14. Yadav, N., Pandey, S., Gupta, A., Dudani, P., Gupta, S., & Rangarajan, K. (2023). Data Privacy in Healthcare: In the Era of Artificial Intelligence. *Indian Dermatology Online Journal*, *14*(6), 788–792. https://doi.org/10.4103/idoj.idoj_543_23

******

Chapter - 10
Implementing AI in Healthcare

Ms. Arti Nagar
Assistant Professor (Department of MCA)
Noida Institute of Engineering & Technology, Greater Noida
Email id-artinagar7379@gmail.com

Dr. Mukta Makhija
Professor & Asst. Dean (IT)-Department of Information Technology
Integrated Academy of Management and Technology (INMANTEC), Ghaziabad
Email ID: asstdean.it@inmantec.edu, mukta.makhija@inmantec.edu

Mr. Neerag kumar (Agarwal)
Computer Application, Assistant Professor
IHM college Mall road Meerut (CCS University)
Email id: agarwneeraj@gmail.com

Mr. Ramachandra Turkani
Assistant Professor (Department of ECE)
KLEIT , Hubli.
Email id -rgturkani@gmail.comr

Abstract
Artificial intelligence (AI) is fundamentally reshaping the healthcare landscape by enabling unprecedented advances in diagnosis, treatment, drug discovery, and patient management. This chapter examines the multifaceted implementation of AI across key healthcare domains, addressing both the remarkable opportunities and the complex challenges that arise. Drawing from a comprehensive body of literature, we explore how machine learning, deep learning, and natural language processing are being deployed in clinical settings worldwide. We further discuss the ethical imperatives, regulatory frameworks, and workforce considerations that must accompany technological adoption. The chapter

concludes with a forward-looking perspective on next-generation AI-driven healthcare systems.

Keywords: artificial intelligence, machine learning, deep learning, clinical decision support, healthcare informatics, medical imaging, electronic health records, drug discovery, ethical AI

1. Introduction

Healthcare has always been a data-intensive discipline. From centuries-old clinical observations to the modern era of genomics and wearable sensors, the volume and complexity of medical information have grown exponentially. Yet the ability of human clinicians to synthesize this data at speed and scale has remained fundamentally constrained by cognitive bandwidth. Artificial intelligence, and in particular machine learning (ML), offers a transformative answer to this challenge.

The term "artificial intelligence in healthcare" encompasses a broad family of computational techniques, including supervised and unsupervised machine learning, deep neural networks, natural language processing (NLP), computer vision, and reinforcement learning. Each of these approaches finds specific and often complementary applications across the clinical enterprise.

Global investment in healthcare AI exceeded $15 billion in 2023, and market analysts project that this figure will surpass $188 billion by 2030, representing a compound annual growth rate of approximately 37% [1]. This explosive growth reflects both the maturation of foundational AI technologies and the acute need for efficiency gains in health systems grappling with aging populations, rising chronic disease burdens, and workforce shortages.

This chapter provides a structured examination of how AI is being implemented across five pivotal domains: (1) medical imaging and diagnostics; (2) clinical decision support; (3) drug discovery and development; (4) personalized medicine and genomics; and (5) hospital operations and administrative efficiency. We then turn to the critical issues of ethics, bias, regulation, and human-AI collaboration before offering a prospective assessment of the field.

2. AI in Medical Imaging and Diagnostics

Medical imaging is arguably the domain in which AI has achieved the most spectacular clinical results to date. Radiology, pathology, dermatology, and ophthalmology all generate vast quantities of visual data amenable to deep learning analysis.

2.1 Radiology and Computed Imaging

Convolutional neural networks (CNNs) trained on large curated datasets have demonstrated radiologist-level performance across a range of imaging tasks. Rajpurkar et al. [2] demonstrated that CheXNet, a 121-layer CNN trained on over 100,000 frontal-view chest X-rays, exceeded the average performance of four practicing radiologists in detecting pneumonia. This landmark study ignited widespread interest and subsequent replication efforts across institutions globally.

In computed tomography (CT) screening, AI algorithms have shown remarkable sensitivity in detecting pulmonary nodules potentially indicative of early-stage lung cancer. The National Lung Screening Trial demonstrated that low-dose CT significantly reduces lung cancer mortality; AI augmentation of this workflow promises to further enhance sensitivity while reducing the false-positive burden that burdened earlier screening protocols [3].

2.2 Digital Pathology

Computational pathology leverages whole-slide imaging (WSI) and deep learning to quantify histopathological features that may be subtle or inconsistent when assessed by the human eye alone. Esteva et al. [4] published a pivotal study demonstrating that a deep convolutional neural network could classify skin lesions with accuracy comparable to board-certified dermatologists, trained on a dataset of 129,450 clinical images. Similar achievements have been reported in the detection of prostate cancer Gleason grading, breast cancer lymph node metastasis, and colorectal polyp classification.

2.3 Ophthalmology

Diabetic retinopathy (DR) represents one of the most clinically significant applications of AI in ophthalmology. Gulshan et al. [5] developed a deep learning algorithm trained on 128,175 retinal images that achieved high sensitivity and specificity for detecting referable DR, matching or surpassing the performance of ophthalmologists. Google's

subsequent collaboration with healthcare providers in Thailand and India demonstrated real-world feasibility, with the AI system operating autonomously in screening settings, dramatically extending the reach of specialist-level eye care to underserved populations.

3. Clinical Decision Support Systems

Clinical decision support systems (CDSS) powered by AI assist clinicians at the point of care by synthesizing patient data and medical knowledge to generate actionable recommendations. Modern AI-driven CDSS represent a substantial evolution beyond earlier rule-based expert systems.

3.1 Early Warning and Sepsis Detection

Sepsis remains one of the leading causes of in-hospital mortality worldwide, with early recognition being the critical determinant of outcome. Machine learning models trained on electronic health record (EHR) data — incorporating vital signs, laboratory values, nursing assessments, and medication records — have demonstrated the ability to identify patients at risk of sepsis hours before clinical deterioration becomes apparent.

Reyna et al. [6] reported results from the PhysioNet/CinC 2019 challenge, in which competing ML models were evaluated on their ability to predict sepsis using real ICU data. Gradient boosting models and recurrent neural networks consistently outperformed simpler scoring tools such as qSOFA and SIRS criteria in terms of early detection sensitivity, though specificity trade-offs remained a practical concern.

3.2 Predictive Analytics and Risk Stratification

Hospital readmission prediction, surgical risk stratification, and mortality forecasting are increasingly powered by ensemble ML models that integrate structured EHR data with unstructured clinical notes processed through NLP. Rajkomar et al. [7] demonstrated that deep learning models trained on EHR data from two large academic medical centers could accurately predict in-hospital mortality, 30-day unplanned readmission, prolonged length of stay, and discharge diagnoses. Notably, these models used raw, minimally preprocessed data, suggesting their robustness for deployment across heterogeneous healthcare systems.

3.3 Medication Safety and Prescribing Assistance

AI-augmented CDSS are being deployed to reduce prescribing errors, flag potential drug-drug interactions, and optimize dosing protocols in real time. Pharmacogenomics-informed AI systems can incorporate a patient's genetic profile to recommend personalized drug selections and dosages, a critical advance in disciplines such as oncology and psychiatry where inter-individual variability in drug metabolism is substantial [8].

4. AI in Drug Discovery and Development

The pharmaceutical industry faces enormous pressure to accelerate drug discovery timelines and reduce attrition rates in clinical development, where over 90% of candidate compounds historically fail to achieve regulatory approval. AI is increasingly central to addressing this challenge across each phase of the drug development pipeline.

4.1 Target Identification and Validation

Large-scale analysis of genomic, proteomic, and metabolomic data using ML enables the identification of disease-relevant molecular targets with greater precision than traditional hypothesis-driven approaches. Graph neural networks (GNNs) are particularly well-suited to modeling complex biological interaction networks, identifying nodes (proteins, genes) whose perturbation is most likely to produce therapeutic benefit [9].

4.2 Molecular Generation and De Novo Design

Generative models, including variational autoencoders (VAEs) and generative adversarial networks (GANs), enable the de novo design of molecules with desired pharmacological properties. Insilico Medicine reported the discovery of a novel fibrosis drug candidate (ISM001-055) within 18 months from target identification to preclinical candidate nomination using an AI-driven platform — a timeline dramatically shorter than the typical 4–6 year industry benchmark [10].

4.3 Protein Structure Prediction: The AlphaFold Revolution

Perhaps no AI achievement in life sciences has generated greater scientific excitement than AlphaFold2, developed by DeepMind. Jumper et al. [11] demonstrated that AlphaFold2 could predict protein three-dimensional structures with atomic-level accuracy, effectively solving a

50-year grand challenge in structural biology. The public release of AlphaFold2 predictions for over 200 million proteins has already catalyzed downstream drug discovery projects across academic and industrial settings, representing a profound democratization of structural biology.

4.4 Clinical Trial Optimization

AI is being applied to optimize clinical trial design, patient recruitment, site selection, and real-time trial monitoring. NLP-powered patient matching algorithms mine EHR data to identify eligible trial participants more efficiently than manual chart review, while predictive models forecast trial completion times and recruitment shortfalls, enabling proactive remediation [12].

5. Personalized Medicine and Genomics

Personalized medicine — tailoring prevention and treatment to the individual characteristics of each patient — represents the philosophical culmination of AI's integration into healthcare. Genomics and multi-omics data provide the molecular substrate upon which AI-driven personalization operates.

5.1 Oncology and Precision Cancer Care

In oncology, tumor molecular profiling using next-generation sequencing (NGS) generates hundreds of thousands of data points per patient. ML algorithms trained on large genomic datasets can identify actionable mutations, predict tumor mutational burden, and stratify patients by likely response to immunotherapy or targeted agents. IBM Watson for Oncology and similar systems demonstrated early promise; however, subsequent analyses revealed important limitations in generalizability and concordance with multidisciplinary tumor board recommendations, underscoring the importance of rigorous validation [13].

5.2 Polygenic Risk Scores

Polygenic risk scores (PRS) aggregate the effects of thousands of common genetic variants to estimate an individual's predisposition to complex diseases such as coronary artery disease, type 2 diabetes, and breast cancer. Khera et al. [14] demonstrated that a genome-wide PRS for coronary artery disease identified a subset of the population with risk

equivalent to monogenic familial hypercholesterolaemia, enabling targeted primary prevention interventions. AI models that integrate PRS with clinical, lifestyle, and imaging data promise further refinement of individual risk estimates.

6. Hospital Operations and Administrative AI

Beyond the clinic and laboratory, AI is delivering substantial value in the administrative and operational domains of healthcare organizations, where inefficiency carries both financial cost and patient safety implications.

6.1 Capacity Management and Patient Flow

Hospital capacity management — predicting demand for beds, operating theaters, and specialist resources — is well-suited to time-series forecasting models. AI-driven demand prediction systems have demonstrated the ability to reduce emergency department boarding times, surgical case cancellations, and ICU bed shortfalls by enabling proactive resource allocation [15].

6.2 Natural Language Processing for Documentation

Physician documentation burden is a major contributor to clinician burnout. Ambient clinical intelligence systems — powered by automatic speech recognition (ASR) and NLP — can generate structured clinical notes from physician-patient conversations in real time, dramatically reducing the time clinicians spend on EHR documentation. Nuance's DAX (Dragon Ambient eXperience) and Amazon's HealthScribe represent leading commercial implementations of this technology [16].

6.3 Revenue Cycle and Fraud Detection

AI-driven automation of medical coding, claims processing, and prior authorization workflows reduces administrative overhead and accelerates revenue cycle velocity. Simultaneously, anomaly detection algorithms trained on claims data can identify patterns consistent with billing fraud, waste, and abuse, with estimated savings of billions of dollars annually for payers and government health programs [17].

7. Ethical, Regulatory, and Workforce Considerations

The implementation of AI in healthcare raises profound ethical, legal, and organizational questions that must be addressed with the same rigor applied to the development of the technologies themselves.

7.1 Algorithmic Bias and Health Equity

A critical risk in healthcare AI is the perpetuation and amplification of existing health disparities through biased training data. Obermeyer et al. [18] published a seminal investigation demonstrating that a commercially deployed risk-stratification algorithm exhibited significant racial bias: because it used healthcare cost as a proxy for health need, it systematically underestimated the illness burden of Black patients, who historically receive less care per unit of illness. This finding prompted widespread calls for bias auditing as a mandatory component of AI validation frameworks.

7.2 Explainability and the "Black Box" Problem

Many high-performing ML models — particularly deep neural networks — are inherently opaque, making it difficult for clinicians to understand why a particular recommendation was generated. Explainability is not merely an academic preference but a clinical necessity: clinicians must be able to critically evaluate AI outputs, identify erroneous recommendations, and maintain accountability for patient care decisions. Techniques such as SHAP (SHapley Additive exPlanations), LIME (Local Interpretable Model-agnostic Explanations), and attention visualization are increasingly incorporated into clinical AI systems to address this challenge [19].

7.3 Regulatory Landscape

Regulatory oversight of healthcare AI varies significantly by jurisdiction. In the United States, the FDA has approved over 900 AI-enabled medical devices as of 2024, with the majority concentrated in radiology. The FDA's proposed framework for AI/ML-based Software as a Medical Device (SaMD) introduces a tiered risk classification approach and provisions for continuous learning systems that update post-deployment [20].

In the European Union, the AI Act (formally adopted in 2024) classifies most healthcare AI applications as high-risk, imposing requirements for transparency, human oversight, accuracy, and robustness, with

conformity assessments required before market placement [21]. Similar regulatory frameworks are evolving in the United Kingdom, Canada, Australia, and Japan, creating a complex multi-jurisdictional compliance landscape for global AI developers.

7.4 Data Privacy and Security

Healthcare data is among the most sensitive categories of personal information. Regulatory instruments such as HIPAA in the United States and GDPR in the European Union impose strict requirements on the collection, storage, use, and sharing of health data. Federated learning — a distributed ML paradigm in which models are trained locally on patient data that never leaves the institution — offers a promising approach to harnessing the power of large multi-institutional datasets without centralizing sensitive information [22].

7.5 Human-AI Collaboration and Workforce Transformation

A nuanced understanding of the optimal division of labor between human clinicians and AI systems is essential for realizing clinical value while preserving patient trust and professional accountability. The emerging consensus in the field conceptualizes AI as an augmentative tool that enhances clinician capabilities rather than a replacement for human judgment and empathy. Tschandl et al. [23] demonstrated that dermatologist performance was significantly improved when clinicians had access to AI assistance compared to either clinicians or AI operating alone, illustrating the power of human-AI teaming.

8. Implementation Challenges and Success Factors

Despite compelling evidence of AI's capabilities in controlled research settings, real-world implementation remains challenging. Understanding the determinants of successful versus failed deployments is essential for healthcare leaders and technology developers alike.

8.1 Data Infrastructure and Interoperability

Effective AI implementation requires access to large, high-quality, well-curated datasets. Healthcare data is notoriously fragmented across incompatible EHR systems, imaging archives, laboratory information systems, and claims databases. The transition to FHIR (Fast Healthcare Interoperability Resources) standards is progressively improving data liquidity, but significant institutional investment in data governance,

master patient indexing, and de-identification pipelines is typically required [24].

8.2 Clinical Integration and Change Management

Technical excellence alone is insufficient for clinical impact. Successful AI implementations require deep co-design with end users, seamless integration into existing clinical workflows, and robust change management programs. Alert fatigue — the desensitization of clinicians to automated alerts due to excessive or poorly calibrated notifications — represents a major barrier to effective CDSS adoption and must be addressed through rigorous alert threshold optimization and user-centered design [25].

9. Future Directions

The trajectory of AI in healthcare over the coming decade promises to be extraordinary. Several emerging developments merit specific attention.

- Large Language Models (LLMs) in Clinical Practice: GPT-4 and successor models have demonstrated impressive performance on medical licensing examinations and clinical reasoning tasks. Their integration as intelligent interfaces for clinician and patient queries, summarization of medical literature, and automated generation of clinical documentation is rapidly accelerating [26].
- Multimodal AI: Next-generation systems will seamlessly integrate imaging, genomics, EHR, wearable sensor, and social determinants data into unified predictive and diagnostic models, approaching a comprehensive digital representation of patient biology and context.
- AI-Enabled Remote and Wearable Monitoring: Continuous remote patient monitoring through AI-analyzed wearable sensor data will increasingly enable early detection of deterioration, management of chronic conditions, and longitudinal population health surveillance outside of traditional care settings [27].
- Digital Twins: Patient-specific computational models that integrate multi-scale biological data with AI simulation capabilities will enable in silico testing of treatment hypotheses prior to clinical application, fundamentally changing the paradigm of personalized treatment planning.

10. Conclusion

Artificial intelligence represents the most consequential technological transformation in healthcare since the advent of evidence-based medicine and the electronic health record. Its potential to democratize access to specialist-level diagnostic capability, accelerate the discovery of life-saving therapeutics, personalize treatment at the molecular level, and relieve overburdened health systems of administrative friction is immense and increasingly well-evidenced.

Yet realizing this potential demands more than technical innovation. It requires a deliberate, multidisciplinary commitment to equity, transparency, rigorous validation, regulatory compliance, and thoughtful human-AI collaboration. The clinician, scientist, health administrator, policymaker, patient, and technologist must each contribute to shaping an AI-enabled healthcare future that is not only more efficient and effective, but more humane.

The journey is underway. The destination — a world in which the right insight reaches the right clinician at the right moment for every patient, regardless of geography or resources — is worth pursuing with both urgency and wisdom.

References

[1] Grand View Research. (2023). Artificial Intelligence in Healthcare Market Size, Share & Trends Analysis Report. Grand View Research, Inc.

[2] Rajpurkar, P., Irvin, J., Ball, R. L., Zhu, K., Yang, B., Mehta, H., ... & Ng, A. Y. (2017). CheXNet: Radiologist-level pneumonia detection on chest X-rays with deep learning. arXiv preprint arXiv:1711.05225.

[3] National Lung Screening Trial Research Team. (2011). Reduced lung-cancer mortality with low-dose computed tomographic screening. New England Journal of Medicine, 365(5), 395–409.

[4] Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., & Thrun, S. (2017). Dermatologist-level classification of skin cancer with deep neural networks. Nature, 542(7639), 115–118.

[5] Gulshan, V., Peng, L., Coram, M., Stumpe, M. C., Wu, D., Narayanaswamy, A., ... & Webster, D. R. (2016). Development and validation of a deep learning algorithm for detection of diabetic

retinopathy in retinal fundus photographs. JAMA, 316(22), 2402–2410.

[6] Reyna, M. A., Josef, C. S., Jeter, R., Shashikumar, S. P., Westover, M. B., Nemati, S., ... & Clifford, G. D. (2020). Early prediction of sepsis from clinical data: The PhysioNet/Computing in Cardiology Challenge 2019. Critical Care Medicine, 48(2), 210–217.

[7] Rajkomar, A., Oren, E., Chen, K., Dai, A. M., Hajaj, N., Hardt, M., ... & Dean, J. (2018). Scalable and accurate deep learning with electronic health records. NPJ Digital Medicine, 1(1), 18.

[8] Roden, D. M., McLeod, H. L., Relling, M. V., Williams, M. S., Mensah, G. A., Peterson, J. F., & Van Driest, S. L. (2019). Pharmacogenomics: The genetics of variable drug responses. Circulation, 123(17), 1854–1872.

[9] Stokes, J. M., Yang, K., Swanson, K., Jin, W., Cubillos-Ruiz, A., Donghia, N. M., ... & Collins, J. J. (2020). A deep learning approach to antibiotic discovery. Cell, 180(4), 688–702.

[10] Ren, F., Aliper, A., Chen, J., Zhao, H., Rao, S., Kuvshinov, A., ... & Zhavoronkov, A. (2023). A small-molecule TNIK inhibitor targets fibrosis in preclinical and clinical models. Nature Biotechnology, 1–12.

[11] Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., ... & Hassabis, D. (2021). Highly accurate protein structure prediction with AlphaFold. Nature, 596(7873), 583–589.

[12] Fogel, D. B. (2018). Factors associated with clinical trials that fail and opportunities for improving the likelihood of success: A review. Contemporary Clinical Trials Communications, 11, 156–164.

[13] Strickland, E. (2017). IBM Watson, heal thyself: How IBM overpromised and underdelivered on AI health care. IEEE Spectrum, 56(4), 24–31.

[14] Khera, A. V., Chaffin, M., Aragam, K. G., Haas, M. E., Roselli, C., Choi, S. H., ... & Kathiresan, S. (2018). Genome-wide polygenic scores for common diseases identify individuals with risk equivalent to monogenic mutations. Nature Genetics, 50(9), 1219–1224.

[15] Poh, M. Z., Swenson, N. C., & Picard, R. W. (2010). A wearable sensor for unobtrusive, long-term assessment of electrodermal activity. IEEE Transactions on Biomedical Engineering, 57(5), 1243–1252.

[16] Sinsky, C. A., & Privitera, M. R. (2018). Creating a manageable cockpit for clinicians: A shared responsibility. JAMA Internal Medicine, 178(6), 741–742.

[17] Centers for Medicare & Medicaid Services. (2022). Fraud Prevention System: Third Report to Congress. U.S. Department of Health and Human Services.

[18] Obermeyer, Z., Powers, B., Vogeli, C., & Mullainathan, S. (2019). Dissecting racial bias in an algorithm used to manage the health of populations. Science, 366(6464), 447–453.

[19] Lundberg, S. M., & Lee, S. I. (2017). A unified approach to interpreting model predictions. Advances in Neural Information Processing Systems, 30.

[20] U.S. Food and Drug Administration. (2021). Artificial Intelligence/Machine Learning (AI/ML)-Based Software as a Medical Device (SaMD) Action Plan. U.S. FDA.

[21] European Parliament and Council of the European Union. (2024). Regulation (EU) 2024/1689 of the European Parliament and of the Council laying down harmonised rules on artificial intelligence (Artificial Intelligence Act). Official Journal of the European Union.

[22] Rieke, N., Hancox, J., Li, W., Milletari, F., Roth, H. R., Albarqouni, S., ... & Cardoso, M. J. (2020). The future of digital health with federated learning. NPJ Digital Medicine, 3(1), 119.

[23] Tschandl, P., Rinner, C., Apalla, Z., Argenziano, G., Codella, N., Halpern, A., ... & Kittler, H. (2020). Human–computer collaboration for skin cancer recognition. Nature Medicine, 26(8), 1229–1234.

[24] Mandl, K. D., & Kohane, I. S. (2016). Time for a patient-driven health information economy? New England Journal of Medicine, 374(3), 205–208.

[25] Ancker, J. S., Edwards, A., Nosal, S., Hauser, D., Mauer, E., & Kaushal, R. (2017). Effects of workload, work complexity, and repeated alerts on alert fatigue in a clinical decision support system. BMC Medical Informatics and Decision Making, 17(1), 36.

[26] Singhal, K., Azizi, S., Tu, T., Mahdavi, S. S., Wei, J., Chung, H. W., ... & Natarajan, V. (2023). Large language models encode clinical knowledge. Nature, 620(7972), 172–180.

[27] Topol, E. J. (2019). High-performance medicine: The convergence of human and artificial intelligence. Nature Medicine, 25(1), 44–56.

ISBN: 978-93-7143-814-8

Chapter - 11
Normalisation in Database Management Systems

Prof (Dr.) Meenakshi Chaturvedi
Professor Dean Research
TIIPS Greater Noida
TIIPS Greater Noida, Affiliated to GGSIP University
Email id-meena10apr@gmail.com

Dr. Vishal Gupta
Assistant Professor (Department-CSE-APEX)
CGC University, Mohali, India
Email id-vishal.gupta8524@gmail.com

Dr. Rupinder Kaur
Assistant Professor (CSE Department)
MMEC, Maharishi Markandeshwar (Deemed to Be University), Mullana, Ambala, Haryana, India.
Email id-rupinder.walia84@gmail.com.

Dr. Sandeep Singh
Assistant Professor (Computer Science Engineering)
SGT University, Gurugram
Email id-Sandeep_soet@sgtuniversity.org

Abstract. Normalisation is a methodical, theory-based procedure to arrange a relational database schema to reduce redundancy of data as well as data update anomalies. The normalisation framework, due to E. F. Codd, was developed in 1970-1972 and operates in a hierarchy of normal forms, including First Normal Form (1NF), the more advanced BoyceCodd Normal Form (BCNF), Fourth Normal Form (4NF), Fifth Normal Form (5NF) and Domain-Key Normal Form (DKNF), based on the motivation of different types of dependencies. In this chapter, the topic of normalisation is rigorously and comprehensively covered based on the functional dependencies, multivalued dependencies, join dependencies and axioms of Armstrong. Covers, decomposition

algorithms, and design trade-offs are discussed thoroughly. An example database schema of a university is worked out using the concept of a case study.

Semantics Keywords normalisation; relational database; functional dependency; normal form; 1NF; 2NF; 3NF; BCNF; 4NF; 5NF; DKNF; multivalued dependency; join dependency; Armstrong axioms; canonical cover; update anomaly; data redundancy; denormalisation; database schema design.

INTRODUCTION

The architectural form of relational database has an overwhelming effect on the accuracy, effectiveness, and sustainability of any information system that relies on the same. An arbitrary graph that contains redundancy of data, i.e. the redundant representation of facts in more than one tuples or relations, is always prone to three imaginations of update anomalies that affect data integrity [1].

- Insertion anomaly: The failure to log a fact and at the same time log a fact that is logically unrelated.
- Deletion anomaly: This occurs in cases when a record is deleted and the information contained in the record is lost.
- Update anomaly: This is the anomaly that occurs when a data item is changed in a single location, and the changes are not reflected in all the copies.

Normalisation eliminates these anomalies by breaking down relations into smaller, well-structured sub-relations by a sequence of transformations which are principled by normal form criteria [2]. The normal forms are defined formally in terms of illegal dependency patterns and all steps in the decomposition are limited to maintain the information content as well as the original constraints. According to Ramakrishnan and Gehrke, normalisation theory offers a strict approach to the analysis and decomposition of relation scheme in a manner that the decomposition is lossless-join and dependency-preserving [3].

This chapter is structured in the following way. Section 5.2 works out the theory of functional dependencies and rules of their inference. In section 5.3 relational keys are discussed. In ascending descending order of

strength, section 5.4 and 5.5 give the major normal forms of 1NF up to DKNF. The fifth section, Section 5.6, is on decomposition properties. Algorithms of normalisation are described in Section 5.7. Section 5.8 poses the issue of denormalisation. In Section 5.9, there is a worked case study, and in Section 5.10 a summary is given.

FUNCTIONAL DEPENDENCIES

The concept of the functional dependency (FD) is the theoretical basis of normalisation. A functional dependency refers to an association between two constituents of attributes in a relation [4].

Formal Definition

Where R is a relation schema and X and Y are non-empty subsets of attributes of R. The functional dependency X - Y is valid in R when and only when, in every instance of a legal relation r(R), in case two tuples t1 and t2 are in agreement on the attributes in X, they must be in agreement on all the attributes of Y [5].

Simply stated, X specializes in defining the value(s) of Y, as the values of Y are defined by X. The Y is the dependent and the X is the determinant. An FD X - Y is trivial if Y [?] X; non-trivial otherwise.

Armstrong's Axioms

The inference rules given by Armstrong are sound and complete to derive all the functional dependencies that are logically implied by a given set of F [6]:

- Reflexivity: If Y [?] X, then X - Y.
- Augmentation: X-Y implies XZ- YZ of any attribute set Z.
- Transitivity: When X- Y and Y-Z, then X-Z.

There are three derived rules which follow:

- Union: If X - Y and X - Z, then X - YZ.
- Decomposition: X -YZ, X -Y, X -Z.
- Pseudo-transitivity: When X -Y and WY-Z, then WX-Z.

Attribute Closure and Canonical Cover

Closure of an attribute set X with regard to F, denoted X + is the largest set of attributes functionally determined by X under F. The most common way of testing whether an FD is already implied by F and finding candidate keys is to compute X + [5].

A canonical cover F is a minimal equivalent of F which has no extraneous attributes on either side of any FD and contains no redundant dependencies. It is computed by: (i) using the union rule to amalgamate FDs that have a common left-hand side; (ii) dropping superfluous attributes; and (iii) dropping redundant FDs. The canonical cover is a basis of the 3NF synthesis algorithm [2].

KEYS IN RELATIONAL SCHEMAS

Understanding of the significant types is a precondition of all the definitions of normal form. There are the following important categories that are common to the relational model [1]:

- Superkey: The combination of attributes the closure of which is the entire set of attributes of R.
- Candidate key: This is a minimal superkey - no proper subset is a superkey.
- Primary key: This key is known as the primary key by the designer and it is the key that identifies the tuple.
- Prime attribute: An attribute used in a candidate key.
- Non-prime attribute: This is any attribute that is not found in any candidate key.

Prime and non-prime attribute are the key concepts behind the formal definitions of 2NF and 3NF. BCNF, in its turn, is only specified in terms of superkeys.

NORMAL FORMS

The first three normal forms were initially presented as foundational papers of 1970 and 1971 [7, 8] by Codd. Normal Form Boyce-Codd proposed the Boyce-Codd Normal Form to remove remaining anomalies which 3NF was unable to resolve. The normal forms of higher degree (4NF, 5NF, DKNF) cover the multivalved and join dependencies [9].

First Normal Form (1NF)

A relation schema R is in the First Normal Form provided and only when the domain of all the attributes in R is atomic that is, consists of indivisible values and has a single value of each attribute in every tuple [7]. First Normal Form prohibits:

- Repeating groups: The sets or lists that are stored in one attribute.
- Multi-valued attributes: an attribute that stores a number of values each of which has a record of a single tuple.
- Composite attributes: structured sub-attributes are lumped into one field.

Table 1 illustrates a relation violating 1NF, followed by its corrected form in Table 2.

StudentID	StudentName	Courses (non-atomic)
S001	Ananya Sharma	DBMS, OS, Networks
S002	Ravi Kumar	Algorithms, OS
S003	Priya Singh	DBMS, Algorithms, Maths

Table 1 A relation violating 1NF: the Courses attribute is non-atomic.

StudentID	StudentName	Course
S001	Ananya Sharma	DBMS
S001	Ananya Sharma	OS
S001	Ananya Sharma	Networks
S002	Ravi Kumar	Algorithms
S002	Ravi Kumar	OS

Table 2 The 1NF-compliant form. Composite primary key: {StudentID, Course}.

Second Normal Form (2NF)

A relation schema R is in Second normal Form and only in 1NF and every non- prime attribute of a relation is fully functionally dependent on every candidate key [1]. A significant dependency Non-prime

attribute A is partially dependency on a candidate key K in such a way that A is functionally determined by some proper subset of K; each such partial dependency is a violation of 2NF.

Example 1 — Partial Dependency
Consider ENROLLMENT(StudentID, CourseID, StudentName, CourseFee, Grade). The composite key is {StudentID, CourseID}. StudentName depends only on StudentID, and CourseFee depends only on CourseID — both are partial dependencies. Only Grade is fully dependent on the composite key. The relation is in 1NF but not 2NF. Decomposition into STUDENT(StudentID, StudentName), COURSE(CourseID, CourseFee), and ENROLLMENT(StudentID, CourseID, Grade) achieves 2NF.

Third Normal Form (3NF)
The relation schema R is said to be in the Third Normal Form, when and only when, all non-trivial functional dependencies of R of the type X - A, either X is a superkey of R, or A is a prime attribute of R [8].

Example 2 — Transitive Dependency
In EMPLOYEE(EmpID, DeptID, DeptName): EmpID → DeptID and DeptID → DeptName yield the transitive dependency EmpID → DeptName. Since DeptName is a non-prime attribute determined by the non-key attribute DeptID, 3NF is violated. Decompose into EMPLOYEE(EmpID, DeptID) and DEPARTMENT(DeptID, DeptName).

The lossless-join, dependency-preserving nNF synthesis algorithm [2] is an important property of 3NF, and it states that it is always possible to synthesize a 3NF out of a dependency-preserving 3NF.

Boyce–Codd Normal Form (BCNF)
A relation schema R is in Boyce-Codd Normal Form, and only if, for all non-trivial X -Y functional dependencies that hold in R, X forms a superkey of R [10]. BCNF is much stronger than the 3NF; all relations in the BCNF are in 3NF but not vice versa.

Example 3 — 3NF but not BCNF
In ADVISING(StudentID, Subject, Advisor): candidate keys are {StudentID, Subject} and {StudentID, Advisor}. The FD Advisor → Subject holds, yet Advisor is not a superkey. This violates BCNF, though Subject is a prime attribute so 3NF is not violated. BCNF decomposition yields ADVISOR_SUBJECT(Advisor, Subject) and STUDENT_ADVISOR(StudentID, Advisor).

The critical limitation of BCNF decomposition is that not all the functional dependencies are always preserved in the decomposition. Relation schema are known in which it is a real design question whether to use a dependency-preserving 3NF decomposition or a non-preserving but anomaly-free BCNF decomposition [3].

Normal Form	Condition Eliminated	Key Requirement	Dep. Preservation Guaranteed
1NF	Non-atomic values; repeating groups	Existence of a primary key	Yes
2NF	Partial dependencies on composite key	Full dependency on entire key	Yes
3NF	Transitive dependencies	Non-prime → via superkey only	Yes
BCNF	All non-trivial FD-based redundancy	Every determinant is a superkey	Not always

Table 3 Comparative summary of normal forms 1NF through BCNF.

Fourth Normal Form (4NF)
To solve the problem of redundancy of multivalued dependencies (MVDs) which are not solved by BCNF, Fagin (1977) introduced the Fourth Normal Form [9]. A multivalued dependency X -Y is a condition that is satisfied by the relation R, meaning that at the event that two tuples coincide about the value of X, the entire set of combinations of Y-values and Z-values should be found in the relation (Z being R [?])A

multivalued dependency X -Y. X [?] Y). A relation may also be said to be in 4NF when X Y is a non-trivial MVD such that X is a superkey [9].

Fifth Normal Form (5NF) and Domain–Key Normal Form (DKNF)

Fifth normal Form (5NF), also called Project-Join Normal Form (PJNF) is used to deal with join dependencies [11]. An R-dependent projection A join dependency {R1, R2,... R[?]} is a join dependency in R when R can be created (losslessly) using those projections. A relation has 5NF as long as all the join dependencies are implied by its candidate keys. Practically, 5NF violations are few and hard to compute.

The ideal, which is proposed by Fagin (1981) isDomain-Key normal form (DKNF): all constraints are the logical derivations of the domain and key constraints only [12]. Although conceptually complete, DKNF remains more or less a theoretical framework because it is very hard to enforce in practice.

DECOMPOSITION PROPERTIES

A relation schema R is decomposed into relation sub-schemas [R1, R2,... R[?]) by any given normalisation step. There are two correctness criteria that have to be fulfilled [3]:

Lossless-Join Decomposition

A R1-R2 decomposition of R is lossless-join if the natural join of R1 and R2 is element-wise the identity, that is, it never speciates extra spurious tuples. The decomposition is lossless when and only when the shared attributes R1 [?] are. R2 is a superkey to either R1 or R2 [5]. Splitting is a property that takes the shape of lossless-join; a lossy decomposition corrupted information.

Dependency-Preserving Decomposition

The property of dependency-preserving A decomposition is a decomposition where the entire system of functional dependencies F can be imposed by verification of constraints in each of the sub-relations, without their combination. This should be true formally, the union of the projections, of F to each R, should be equivalent to F [2]. Under these circumstances, some integrity conditions can no longer hold and it depends on costly join operations to enforce the integrity.

NORMALISATION ALGORITHMS

The 3NF Synthesis Algorithm

The 3NF synthesis algorithm of Bernstein is a dependency-preservation decomposition to 3NF that is lossless-join [2]:

1. Word count compute the canonical set cover F of F.
2. Given the FD X - Y of F, form a relation schema R = X [?]. Y.
3. In case a schema generated in Step 2 does not include a candidate key of R, include a schema that involves any candidate key.
4. Eliminate any schema whose values are a subset of those in another schema(redundant removal).

The BCNF Decomposition Algorithm

BCNF decomposition algorithm generates lossless-join decomposition which might not be able to produce all dependencies [3]:

5. Start with the schema set R 0 = O.
6. Otherwise, assume that all the schema in O is in BCNF, stop.
7. Select a schema R [?] O not in BCNF. Let X -Y a violating FD (X is not a superkey of R).
8. Replace R in O with R1 = X [?] Y and R2 = R [?] Y [?] X. Return to Step 2.

DENORMALISATION AND PRACTICAL CONSIDERATIONS

Very normalised schemas can be degraded in read intensive applications through multi-table joins [13]. Denormalisation is the intentional insertion of controlled redundancy to enhance read throughput, which is usually in the data warehousing and OLAP scenarios. Common strategies include:

- Table merging: This method combines tables that frequently share common rows so that the cost of joining two tables is removed.
- Derived attributes: It stores pre-calculated or aggregate values in order to eliminate repetitive calculation.

- Star and snowflake schemas: Data warehousing paradigms of dimensional modelling [14].

Denormalisation decision should be well-informed by the evaluation of workload profile, frequency of updates and permissible consistency trade-offs. A thorough theoretical treatment of such balance is provided by date [15].

WORKED CASE STUDY: UNIVERSITY DATABASE

An example of unnormalised relation that contains student enrolment information is as follows:

UNIVERSITY(StudentID, StudentName, Email, CourseID, CourseName, Credits, InstructorID, InstructorName, Dept, Grade, EnrolDate)

The functional dependencies given in this schema:

- StudentID StudentName, Email.
- CourseID-CourseName, Credits, InstructorID
- InstructorID - instructor Name, department.
- StudentID, courseID - Grade, date of enrolment.

Step 1 -- 1NF. Attributes are all atomic and the relation is in 1NF already.

Step 2 -- 2NF. The composite key StudentID, CourseID admits partial dependency Studentname and email is dependent on StudentID StudentCredits and instructorID on CourseID. Decompose into:

- **Where StudentID, StudentName and Email are data types.**
- **COURSE(CourseID, CourseName, Credits, InstructorID)**
- **ENROLLment(StudentID, CourseID, Grade, EnrolDate)**

Step 3 — 3NF. In COURSE: CourseID → InstructorID and InstructorID → InstructorName, Dept constitute a transitive dependency. Extract a separate relation:

- INSTRUCTOR(InstructorID, InstructorName, Dept)

- COURSE revised: COURSE(CourseID, CourseName, Credits, InstructorID)

Step 4 -- BCNF. Make sure that all the determinants on all relation are superkeys. The four last schemas meet the requirement of BCNF.

Relation	Primary Key	Foreign Keys	Normal Form
STUDENT	StudentID	—	BCNF
INSTRUCTOR	InstructorID	—	BCNF
COURSE	CourseID	InstructorID → INSTRUCTOR	BCNF
ENROLLMENT	{StudentID, CourseID}	StudentID → STUDENT; CourseID → COURSE	BCNF

Table 4 Final normalised university database schema with key and referential constraints.

KEY TERMS

Normalisation The algorithm used to create a relational schema to minimise redundancy of data and to remove update anomalies by enforcing normal form properties.

Functional Dependency (FD) X -Y: X is the sole determiner of Y in every legal relation instance.

Trivial FD An FD X - Y where Y [?] X; is vacuously in all relations.

Partial Dependency This is a non-prime attribute that is defined by a proper subset of a composite key; it does not satisfy 2NF.

Transitive Dependency A non-prime FD X - Z through an intermediate (non-prime) attribute Y (X - Y, Y - Z); is a 3NF violation.

Multivalued Dependency (MVD) X Y: Y-values of the X are independent of all other attributes. Addressed by 4NF.

Join Dependency A relation is the lossless join of given projections. Addressed by 5NF.

Axioms Axioms Axioms Formulated to derive all implied FDs using a sound, complete set of three inference rules (reflexivity, augmentation, transitivity).

Attributes Closure (X+) The attributes that are determinable in X given a set of FDs F.
Canonical Cover (F) An equivalent of the minimal, non-redundant form of F, which is used in the 3NF synthesis algorithm.
Superkey Any attribute set the closure of which is equal to the entire attribute set of R.
Candidate Key Minimal superkey.
Prime Attribute An attribute of one of the candidate keys.
Non-Prime Attribute An attribute that is not a part of a candidate key.
First Normal Form (1NF) All the attributes are atomic domains; no repeating groups or multi-valued attributes.
Second normal form (2NF) 1NF and no partial dependencies between non-prime attributes and candidate keys.
Third Normal Form (3NF) 2NF and no non-prime dependency transitivity on candidate keys.
BoyceCodd normal Form (BCNF) All determinants of all non trivial FDs are super keys.
Fourth Normal Form (4NF) BCNF and no non-trivial non-trivial MVDs of whose left-hand side is not a superkey.
Fifth normal form (5NF) 4NF and all the join dependencies are implied by candidate keys.
Domain-Key Normal Form (DKNF) All constraints are logical extensions of domain and key constraints; the hypothetical extreme of normalisation.
Lossless-Join Decomposition A decomposition where the natural join of sub-relations is the same relation with no spurious tuples.
Dependency-Preserving Decomposition A decomposition where F, the original FD set, is enforceable in the individual (sub-relations without joins) sub-relations.
Update Anomaly Any inconsistency that is created due to uncontrolled data redundancy, that is, insertion, deletion, or modification.
Denormalisation Introduced redundancy to enhance the performance of read queries, usually found in systems of analysis.

SUMMARY

This chapter has given an extensive coverage on the concept of normalisation in the design of relational database, which has explored

the topic of design and its practical implementation taking into consideration all the theoretical components of the concept.

We started by encouraging normalisation using the three anomalies of updates, the insertion, the deletion, and the modification that are a result of data redundancy. Then the formal apparatus of the functional dependencies were elaborated, which comprised of the axioms of Armstrong, the attribute closure, and the canonical covers. The standard key types were defined as superkey, candidate key, prime and non-prime attribute and defined strictly as conditions that were required before the normal form conditions.

The hierarchy of normal forms was introduced in the strictness of elimination, 1NF non-atomic values, 2NF partial dependencies, 3NF transitive dependencies, BCNF every FD determinant is a superkey, 4NF non-multivalued dependencies, 5NF join dependencies, and DKNF the theoretical ideal. The two properties crucial to correctness of any valid decomposition were found to be the lossless-join and dependency preserving properties and both the 3NF synthesis algorithm and the BCNF decomposition algorithm were described.

The chapter then discussed and concluded with a trade-offs discussion on the denormalisation and a worked case study of a full normalisation of a university schema, transforming a design in an unnormalised universal relation to a BCNF compliant set of four relations which are well structured.

REFERENCES

1 Silberschatz, A., Korth, H. F., and Sudarshan, S. (2019) Database System Concepts, 7th edn. McGraw-Hill Education, New York.

2 Bernstein, P. A. (1976) Synthesizing third normal form relations from functional dependencies. ACM Transactions on Database Systems, 1(4), 277–298.

3 Ramakrishnan, R. and Gehrke, J. (2003) Database Management Systems, 3rd edn. McGraw-Hill, New York.

4 Ullman, J. D. and Widom, J. (2008) A First Course in Database Systems, 3rd edn. Pearson Prentice Hall, Upper Saddle River, NJ.

5 Maier, D. (1983) The Theory of Relational Databases. Computer Science Press, Rockville, MD.

6 Armstrong, W. W. (1974) Dependency structures of data base relationships. Proceedings of IFIP Congress 74, pp. 580–583. North-Holland, Amsterdam.

7 Codd, E. F. (1970) A relational model of data for large shared data banks. Communications of the ACM, 13(6), 377–387.

8 Codd, E. F. (1971) Further normalization of the data base relational model. IBM Research Report RJ909. IBM Corporation, San Jose, CA.

9 Fagin, R. (1977) Multivalued dependencies and a new normal form for relational databases. ACM Transactions on Database Systems, 2(3), 262–278.

10 Boyce, R. F. and Codd, E. F. (1974) Relational completeness of data base sublanguages. In: Rustin, R. (ed.) Data Base Systems, pp. 65–98. Prentice-Hall, Englewood Cliffs, NJ.

11 Fagin, R. (1979) Normal forms and relational database operators. Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data, pp. 153–160. ACM, New York.

12 Fagin, R. (1981) A normal form for relational databases that is based on domains and keys. ACM Transactions on Database Systems, 6(3), 387–415.

13 Date, C. J. (2003) An Introduction to Database Systems, 8th edn. Addison-Wesley, Boston, MA.

14 Kimball, R. and Ross, M. (2013) The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling, 3rd edn. Wiley, Hoboken, NJ.

15 Date, C. J. (2012) Database Design and Relational Theory: Normal Forms and All That Jazz. O'Reilly Media, Sebastopol, CA.

16 Elmasri, R. and Navathe, S. B. (2015) Fundamentals of Database Systems, 7th edn. Pearson, Hoboken, NJ.

****